Azure Resource Manager (ARM) is the native platform for [infrastructure as code](https://cloud.netapp.com/blog/save-time-and-headaches-with-infrastructure-as-code-on-azure-cvo-blg)[(IaC) in Azure](https://cloud.netapp.com/blog/save-time-and-headaches-with-infrastructure-as-code-on-azure-cvo-blg). It enables you to centralize the management, deployment, and security of Azure resources. ARM groups resources into containers that group Azure assets together. You can use ARM to deploy assets from multiple Azure resource provider services, such as Microsoft Storage and Microsoft Compute.

In this post, we’ll examine key concepts and benefits of Azure Resource Manager and explain three best practices you should implement when adopting ARM templates. We’ll also explain how [NetApp Cloud Volumes ONTAP](https://cloud.netapp.com/ontap-cloud) can help simplify storage when using infrastructure as code in Azure.

**In this article, you will learn:**

* [**What is Azure Resource Manager (ARM)**](https://cloud.netapp.com/blog/azure-cvo-blg-azure-resource-manager-arm-benefits-and-best-practices#H_H1)
* [**Benefits of Azure Resource Manager**](https://cloud.netapp.com/blog/azure-cvo-blg-azure-resource-manager-arm-benefits-and-best-practices#H_H2)
* [**ARM templates best practices**](https://cloud.netapp.com/blog/azure-cvo-blg-azure-resource-manager-arm-benefits-and-best-practices#H_H3)
* [**Azure Resource Manager with Cloud Volumes ONTAP**](https://cloud.netapp.com/blog/azure-cvo-blg-azure-resource-manager-arm-benefits-and-best-practices#H_H4)

**What Is Azure Resource Manager?**

Azure Resource Manager (ARM) is an Azure service you can use to manage and deploy resources [using an infrastructure as code](https://cloud.netapp.com/blog/why-you-need-infrastructure-as-code-to-do-devops-properly-cvo-blg) paradigm. It enables you to provision, modify, and delete resources using a variety of features including access controls, tags, and locks.

When using Azure Resource Manager, there is some specific terminology you should be aware of. The most common terms include:

* **resource**—an asset that can be managed. Assets include virtual machines (VMs), virtual networks, databases, web apps, and storage accounts. Resources may also refer to tags, subscriptions, resource groups, or management groups.
* **resource group**—a container that groups together related resources. An Azure resource group enables you to manage multiple resources as a whole.
* **resource provider**—an individual service in Azure that you can create resources in. For example, Microsoft Storage or Microsoft Compute.
* **Resource Manager template**—a file that defines how resources should be deployed to groups, subscriptions or tenants. This file is defined in JavaScript Object Notation (JSON).
* **declarative syntax**—the syntax used for Resource Manager templates. It enables you to define how resources should be handled without having to know programming commands.

**Key Benefits of Azure Resource Manager**

Azure Resources Manager provides several benefits to ease resource management and deployment. These benefits include the following.

**Centralized management**

ARM enables you to manage resources from a centralized dashboard. It enables you to integrate any Azure resources you are using, including third-party applications and services adopted from the Microsoft Azure Marketplace.

This centralization makes it easier to ensure that resources are handled consistently. It also increases visibility into the configuration of your resources and enables you to restrict administrative workflows to a single tool.

**Templates**

Azure Resource Manager templates enable you to define and deploy any Azure resources you need to in a clear and consistent way. ARM templates enable you to define resource details, including names, locations, availability zones, security settings, and networks. Once defined, you can use Azure templates repeatedly to deploy identical resources, as needed for distributed applications, testing, or development.

ARM templates are customizable and can be built or modified through Visual Studio Code, Visual Studio, or the Azure Portal. You can also fully automate ARM template deployment, removing dependencies on manual configurations. Additionally, if you are using third-party tools, Azure Marketplace solutions often provide readily available ARM templates.

**Security**

ARM enables you to use a wide variety of security features with your resources, including role-based access controls (RBACs) and OAuth authentication. It provides granular control over resources and enables you to restrict who is able to modify which workloads. This includes the ability to lock resources against modification or deletion.

Additionally, within Azure Resource Manager, all events are logged with event changes and users. This enables you to audit management as needed and helps ensure that your environment management meets compliance standards.

**ARM Template Best Practices**

When creating ARM templates there are several best practices you should keep in mind. These practices can help make management easier and ensure that your resources are configured efficiently.

**Template limits**

In general, try to limit your templates to 4MB with parameters no larger than 64KB. These limits should be applied to the size of your final template, including any iterative resource definitions, variables, or parameters you need to include.

Additionally, you should keep in mind that there are some hard limits for templates. These restrict you to:

* 24,576 characters in a template expression
* 64 output values
* 800 resources (including copy count)
* 256 variables
* 256 parameters

**Resource groups**

Any resource groups you create store metadata about your resources. This metadata is stored in the same location as your group. This means that if a group region is inaccessible, you cannot update or manage those resources until region access is restored.

If you need to create a failover for a resource group, be sure to keep a copy of your metadata in the failover region. This ensures that you retain control even if your original group is inaccessible.

**Consider YAML**

The format for ARM templates is JSON. This format is fine for simple templates but as resource numbers and deployment complexities scale, these templates lose their human-readability. For example, some Azure resources require over a hundred lines to define with seemingly endless nested brackets. This makes interpreting the template definitions very difficult.

While you cannot change this format, you can write your templates in YAML and convert to JSON when ready to deploy. YAML can increase the readability of your templates and enable you to comment why you are defining resources as you are. This ability to comment is especially helpful if you have multiple team members working to define templates.

**Avoid linked templates unless necessary**

Linked templates are a feature in Azure Resource Manager that enable you to split your resources into individual templates. These templates are then linked together and tied to a single deployment source.

This linking is great except that it requires templates to either be publicly accessible or that your deployment pipeline is accessible. You can provide this later access with a SAS token but this still adds some vulnerability to your systems.

Additionally, when using linked templates, you are restricted to using either parameter URI or parameter objects. You cannot combine formats, restricting your ability to handle sensitive information inside the template.

Rather than trying to work around these issues, you should consider using standard templates. You can use your pipelines to control deployment and ensure that dependent templates are deployed together. This enables you to achieve the same result as linked templates with greater security.

For more tips about efficient cloud optimization, see our article about [Azure quickstart templates](https://cloud.netapp.com/blog/azure-cvo-blg-azure-quickstart-templates-how-to).

**Azure Resource Manager with Cloud Volumes ONTAP**

NetApp [Cloud Volumes ONTAP](https://cloud.netapp.com/ontap-cloud), the leading enterprise-grade storage management solution, delivers secure, proven storage management services on AWS, Azure and Google Cloud. Cloud Volumes ONTAP supports up to a capacity of 368TB, and supports various use cases such as file services, databases, DevOps or any other enterprise workload, with a strong set of features including high availability, data protection, storage efficiencies, Kubernetes integration, and more.

In particular, Cloud Volumes ONTAP provides [Cloud Manager](https://cloud.netapp.com/cloud-manager), a UI and APIs for management, automation and orchestration, supporting hybrid & multi-cloud architectures, and letting you treat pools of storage as one element in your infrastructure as code setup