Adopting Azure Resource Manager for efficient cloud infrastructure management

Microsoft IT runs in the cloud. We have a massive investment in Microsoft Azure for our cloud IT infrastructure, and we are continually looking for ways to deploy and manage Azure resources in the most efficient way possible. Following the public implementation of the Azure Resource Manager model for Azure infrastructure as a service (IaaS) and platform as a service (PaaS) components, we have been moving our cloud infrastructure from the Azure Service Management model to the Azure Resource Manager model. During this transition, we learned a lot, which has helped us achieve the greatest possible benefit from the new model.

Making Azure the first choice for IT infrastructure

The cloud-first, mobile-first culture at Microsoft is designed to give our business groups the most effective IT environment possible. For our IT teams, this means being able to create that environment quickly, to the required scale, and in a cost-effective manner. Microsoft has championed a move to the cloud because it gives us the infrastructure we need to power the next generation of business applications. It also elevates collaboration and productivity, enabling our employees to be more successful.

Microsoft Azure is at the core of our cloud infrastructure. We are continually moving applications to the cloud, and Azure is the first choice for new IT solutions that we implement. We currently support the largest public, cloud-based corporate IT infrastructure in the world using Azure. Our Azure environment includes:

- More than 2,400 active Azure subscriptions.
- More than 200 cloud-based applications.
- More than 12,000 Azure virtual machines.
- More than 18 billion Azure Active Directory authentications per week.
- More than 30 trillion objects stored on Microsoft Azure.

Our business runs on Azure, and we are dedicated to increasing our footprint in the cloud, migrating our on-premises resources to Azure. The sheer volume of infrastructure in Azure requires a management solution and approach that allows our IT teams to deploy and manage Azure resources in an efficient and timely manner.

Understanding Azure Resource Manager

Azure Resource Manager provides the latest framework for the resources used to create solutions in Azure. It gives you a new way to deploy and manage Azure resources as a solution. For example, for an application that is used to track sales records your solution might consist of several Azure virtual machines connected by Azure Virtual Networks, and using a variety of other Azure resources.

The Azure Resource Manager model involves several important components:

- **Resource groups.** A resource group is a container that holds related resources for an application. A resource group can include all of your resources for an application or only those resources that are logically grouped together. You can decide how you want to allocate resources to resource groups based on what makes the most sense for your organization.
• **Role-based access control.** You can add users to predefined platform and resource-specific roles and apply those roles to a subscription, resource group, or resource to limit access. For example, you can use the predefined role known as SQL DB Contributor that lets people manage databases, but not database servers or security policies. You add people in your organization that need this type of access to the SQL DB Contributor role and apply the role to the subscription, resource group, or resource.

• **Tags.** You can use tags to categorize resources according to your managing or billing requirements. You might want to use tags when you have a complex collection of resource groups and resources that you need to visualize in the way that makes the most sense to you. For example, you can tag resources that serve a similar role in your organization or that belong to the same department. Without tags, people in your organization can create multiple resources that may be very difficult to identify and manage later. For example, you may want to delete all of the resources for a particular project, but if those resources were not tagged for the project, you will have to manually find them. Tagging can be an important way for you to reduce unnecessary costs in your subscription, and tags exist independently of resource groups in Azure.

• **Templates.** Templates define the deployment and configuration of your application. Templates provide a declarative way to define deployment. By using a template, you can repeatedly deploy your application throughout the app lifecycle and have confidence that your resources are deployed in a consistent state.

• **Policy.** You create policy definitions that describe the actions or resources that are specifically denied in Azure. You assign those policy definitions at the desired scope, such as the subscription, resource group, or an individual resource. For example, you can use policy to control where resources are allowed to be created. Or you might want to control access to the resources by allowing only certain types of resources to be provisioned.

Using these components, the Azure Resource Manager model provides several advantages over the previous infrastructure model, Azure Service Management:

• You can deploy, manage, and monitor all of the resources for your solution as a group, rather than individually handling these resources.

• You can repeatedly deploy your solution throughout the development lifecycle and have confidence that your resources are deployed in a consistent state.

• You can use declarative templates to define your deployment.

• You can define the dependencies between resources so that they are deployed in the correct order.

• You can apply access control to all services in your resource group because role-based access control (RBAC) is natively integrated into the management platform.

• You can apply tags to resources to logically organize all of the resources in your subscription.

• You can clarify billing for your organization by viewing the rolled-up costs for the entire group or for a group of resources that share the same tag.

### Enabling agile cloud infrastructure

Our implementation of Azure Resource Manager started with new implementations of Azure solutions. We adopted Azure Resource Manager as the default model for all new solutions. New solutions transitioned smoothly to Azure Resource Manager, and most of our existing Azure infrastructure remained in the Azure Service Management model. However, there were cases where we needed a solution to access resources from both models, and we also had to reconcile the way we monitored and managed resources from each model to provide a monitoring and management environment that is as unified as possible.

### Integrating Azure Service Management and Azure Resource Manager

Although the requirement for integration was not universal, it did have a significant impact on our applications in Azure, and it necessitated—at the very least—communication between resources. The primary method for enabling communication between Azure Service Management virtual machines and Azure Resource Manager virtual machines was creating a virtual network infrastructure that could support both models. Virtual network gateways were created on both the Azure Service Management and Azure Resource Manager virtual networks, and the gateways were joined to facilitate network traffic between virtual machines and other components on each of the respective networks.
Reassessing and retooling
The APIs and associated management tools are mostly mutually exclusive, so we operated in two separate management spaces using two sets of tools. Our Azure Service Management solutions continued to use the Azure Service Management-based tools, whereas our Azure Resource Manager infrastructure was managed by a new and growing set of tools designed for Azure Resource Manager.

Certain aspects of our management environment—such as our change management database—managed both platforms and held information for both Azure Service Management and Azure Resource Manager components, but there weren’t common tools to directly manage the infrastructure. Azure Service Management-based tools that we wanted to use with Azure Resource Manager components were rewritten using the Azure Resource Manager APIs, and both tools maintained independent development and lifecycles.

Using templates to simplify deployment and management
Azure Resource Manager templates give you the opportunity to create standardized deployment and management processes that are reusable within the Azure Resource Manager environment. We used templates heavily throughout the Azure Resource Manager deployment process.

To store and deploy the templates, we used a GIT repository that publishes automatically to a GitHub repository, which is available externally. We used nested or linked templates that require access to a publicly accessible URI—in this case, GitHub—specifically, raw.githubusercontent.com. These repositories gave us the ability to store our templates centrally, to establish standards, and to provide a location where people could start their own deployments. By using these repositories, people were able to clone their own instance of the repository to their local machine for customization, modification, and testing before publishing changes. GitHub also gave us the ability to take advantage of our internal community contributions to the Azure Resource Manager template ecosystem. The two repositories that we used were:

- **Internal Azure Resource Manager Repository**. This is where we stored our standard, compliant, and parallel Azure Resource Manager templates that we support. These templates allowed us to specialize deployment for scenarios such as domain-joined virtual machines, standardize configuration of drives and storage, and create and implement common Azure tags on deployed resources.

- **Community Azure Resource Manager Repository**. This repository was open for internal Microsoft use, but we did not validate or check compliance. This repository allowed people to create their own templates and share them with the rest of the organization so that innovation and Azure Resource Manager development could be shared. This repository does not publish to GitHub.

Best practices for Azure Resource Manager implementation
We established several design and implementation practices that helped us to create a consistent and uniform Azure Resource Manager environment. With these best practices, we realized significant efficiencies in the deployment and management of Azure Resource Manager–based resources.

Delineate services using subscriptions
We established that Azure subscriptions would be created for each service and named accordingly. This created a very detailed level of control over Azure resources. At Microsoft, a service is defined as a primary business function. There are currently more than 200 services, which often contain multiple applications and resource groups. We established a services naming convention to provide consistency and continuity. In addition to the name of the service, our naming convention included using the company prefix of “MSFT” and a numeric suffix to allow multiple subscriptions for the same service if it ever becomes necessary. For example:

- MSFT Customer Relationship Management 1
- MSFT International Tax 1
- MSFT Commercial Business Reporting 2
- MSFT Network and Infrastructure Mgmt IT Solutions 3
Use resource groups effectively
We recommended that applications have their own dedicated resource group. Resource groups are created within each subscription, and each resource group contains objects or applications that share a similar lifecycle. Resource groups can have specific RBACs assigned—each resource group will have different users assigned different permissions. Virtual machines in a solution must be members of a single resource group, and there will often be multiple virtual machines in each resource group.

Use RBAC to establish permissions for groups
In our Azure environment, RBAC is designed to establish role permissions for resource groups or subscriptions. We used our Active Directory environment and synchronized with groups in RBAC on Azure. We used the built-in roles to define permission levels:

- **Reader**: Can view all Azure components except secrets, but can’t make changes.
- **Contributor**: Can manage everything except user access.
- **User Access Administrator**: Can manage user access to Azure resources.

We then used those roles to establish permissions for our internal RBAC groups.

Use tags for flexible resource management
Using tags provides easy management across many scenarios. Tags were helpful in grouping resources for scenarios outside of the scope of typical application maintenance. We reserved a certain number of tags so that resources could be easily identified by change management database users. We also used tags extensively for tracking compliance and billing, which was useful when compliance or billing-related resources spanned multiple applications or resource groups.

Use templates to deploy and manage consistent solutions
Templates were an immense change to the way we created and managed Azure resources. By using the JavaScript Object Notation (JSON) templates in Azure Resource Manager, we deployed and manage an entire solution—including virtual machines, virtual networks, and associated resources. By using templates, we defined how and when Azure resources were created or changed. We were also able to create dependencies and implementation requirements for complex applications with large, complex sets of Azure resources. Templates are hosted on GitHub, so people can innovate and develop templates.

Moving forward
We have integrated Azure Service Management and Azure Resource Manager. However, our goal is to migrate our on-premises and Azure Service Management resources to the Azure Resource Manager model in the cloud. We are working on two goals for this migration:

- **Use Azure Site Recovery** to move virtual machines from on-premises datacenters into Azure Resource Manager, from Azure Service Management to Azure Resource Manager, and from Azure Resource Manager to Azure Resource Manager across subscriptions. This method allows teams to do the work themselves, using internal documentation that we create. This means that they can do their own development and operations, and move virtual machines on their own schedule.
- **Test a set of Microsoft-internal tools** for migrating a full subscription from Azure Service Management to Azure Resource Manager. We’ve been providing feedback about these tools to our product group, which is actively developing the migration solution. This method requires us to move a whole subscription at once, which means coordinating with many teams that might be involved.