Automating cloud infrastructure management with Azure Resource Manager

Core Services Engineering (CSE, formerly Microsoft IT) is responsible for managing and implementing its cloud infrastructure. An internal team has been working toward effective automation and management across the cloud environment, creating a set of tools and standards to help Microsoft maintain an effective and efficient cloud presence.

In line with the Microsoft vision to be cloud-first, mobile first, the CSE cloud infrastructure hosts most Microsoft apps and services. We also provide most of the business functionality for Microsoft business groups, including infrastructure across both private cloud and public cloud environments, Hyper-V, System Center, and Microsoft Azure. The CSE cloud environment consists of:

- Almost 2,500 Hyper-V hosts.
- More than 15,000 Hyper-V virtual machines.
- More than 1,800 Azure virtual networks.
- Almost 19,000 Azure virtual machines.
- More than 600 Azure platform as a service (PaaS) apps.
- More than 5,000 Azure SQL instances.

Assessing cloud management

As cloud adoption has grown at Microsoft, we have been challenged to create management tools and frameworks that enable us to provide an environment in which users receive the best experience with cloud platform apps and services. We found that the management environment for our cloud environment was becoming more and more fractured. Different business groups were using different tools for deployment. Some tools included built-in automation and some involved manual creation and configuration of cloud components, like virtual machines and virtual networks. The environment caused several issues that we wanted to remedy:

- Cloud IT operations were being performed in inconsistent ways. No broad standards had been established across business groups for deploying and configuring cloud infrastructure.
- Manual processes allowed for human error and inconsistent configuration. Manually entered data was not entered in a consistent format. Free-form text fields in the creation process provided opportunities for misnamed objects.
- Traditional IT provisioning processes required approval or tasks to be performed by many different teams, which added time to deployment tasks.

In addition to the shortcomings of the current environment, we also recognized the opportunity to take advantage of the benefits of Azure Resource Manager. It provided a more modular and declarative method for deploying and managing Azure infrastructure as a service (IaaS) components such as virtual machines and virtual networks. Azure Resource Manager had become the default platform for new cloud infrastructure deployed in Azure—and many of our tools did not fully support it.
Planning for consistent deployment and management using automation

We wanted to remove as many opportunities as possible for human error or other events that caused misconfigured cloud infrastructure. Automation provided the ability to remove human input that could cause errors. It also created distinct deployment and management tasks that would provide a consistent experience for anyone deploying or managing cloud infrastructure. Although many of our processes were performed by several different people, we wanted the automated solution to require as little input from our team as possible. Self-service was the ultimate high-level goal. We also established several other goals:

- **Streamline deployment and management tasks.** Remove as many manual touches as possible in all of our processes and reduce the number of people or teams that are involved in deployment and management processes.

- **Provide a consistently configured environment.** We wanted automation to provide configuration procedures that would result in consistent naming conventions for components and objects across the CSE cloud infrastructure environment.

- **Reuse data within the environment.** Many of our pre-existing tools and processes involved a lot of manual entry of names and variables. In many cases, these names and variables already existed in the cloud infrastructure environment. We wanted our solution to have access to that data and re-use it instead of requiring it to be re-entered each time a virtual machine or virtual network was created.

- **Support Azure Resource Manager as the default model for creating Azure IaaS components.** Azure Resource Manager is the future of Azure IaaS, so we wanted to make sure that any new Azure deployments were using the most current IaaS model.

Enabling automation in cloud infrastructure

Because of the widespread nature of our pre-existing processes for cloud management, we needed to establish a plan and a foundation for a new solution. The new solution needed to satisfy the requirements of all of the teams involved—yet be robust and modular enough that all of the teams could use the same toolset.

Creating the building blocks for modular automation and infrastructure as code

We started building our solution in Azure Resource Manager because it was used to create the bulk of our new resources. It also provided the best toolset for modularity, consistency, and repeatability. With Azure Resource Manager, we have started to implement infrastructure as code—where the same, smaller building blocks of Azure deployment code can be used in many scenarios to create infrastructure in Azure. We created a set of Azure Resource Manager tools for deployment in the following categories:

- **Templates.** Azure Resource Manager templates provided a method to create consistency throughout our deployment and management processes. The JavaScript Object Notation (JSON) files provided a standard within which we could define resources and resource groups involved in a deployment. By using templates, we were also able to define dependencies between components and reuse data between across deployment and management processes.

- **Scripts.** PowerShell scripts provided the engine of our automation capability. A lot of the deployment and management logic was built into PowerShell scripts. In turn, these scripts used the templates we created to define our resources.

- **Interfaces.** We created web interfaces to provide a more user-friendly and consistent environment within which to run and manage scripts and templates. We also used APIs to interface with apps.
We also created tools for the Azure Service Management platform. Some of our cloud resources were still in Azure Service Management, so we created tools to ensure that our cloud environment was as consistent as possible across both platforms. We were able to reuse some components between both the Azure Resource Manager platform and the Azure Service Management platform, but many scripts needed to be created separately in each platform.

**Automating for optimization**

A significant factor in our Azure environment was the need for optimization. We wanted to be able to streamline our optimization process to use the most appropriate amount of compute resources possible. Many of our apps and services hosted in Azure experience load fluctuations or aren’t required to run around the clock. We created a framework in Azure that enabled us to dynamically scale our compute cores based on demand. This included scaling based on the actual compute cycles required, and it allowed us to shut down compute cores when they weren’t required because of scheduling parameters.

**Automation in practice: creating Azure virtual machines**

The creation of Azure virtual machines within our automation solution provides a practical look at how we implement automation and how the different pieces fit together to provide an effective solution. Creating an Azure virtual machine involves three primary components:

- **Templates.** We provide a set of Azure Resource Manager templates that correspond with the most common virtual machine configurations in our environment. Templates are available through a CSE-curated catalog hosted on GitHub. The templates contain standard deployment practices. They ensure that the resulting virtual machine contains the necessary service accounts and Desired State Configuration (DSC) definitions for Windows configuration within the virtual machine.

- **Azure Resource Manager Policy.** We used Azure Resource Manager policy to control certain aspects of resource deployment, such as naming conventions and locations where resources can be created. This policy helped us to remove potential for human error and create a cleaner deployment process.

- **DSC.** DSC enables configuration of Windows components and applications within a virtual machine. The definitions for DSC functionality are most often defined within the template, but they can also be defined with a PowerShell script and managed with Operations Management Suite. DSC enables the automation process for end-to-end configuration of the virtual machine without manual intervention.

- **PowerShell.** The various Azure modules for PowerShell provide the core of the automation functionality. By using it, we incorporated all of the required deployment and configuration tasks to create a virtual machine into a single PowerShell script.

- **Group Policy.** Group Policy provides additional configuration capabilities for Windows-based virtual machines joined to the domain. By using Group Policy, we could further configure the operating system and user environment within a virtual machine.

**Automating toward flexible networking**

We found that changing the network infrastructure required a lot of manual intervention, both virtual and physical. Our approach to automating networking was primarily focused on removing the necessity to make changes to the physical network, whether that meant physical port changes, reconfiguring wiring, or adding and removing physical equipment.

We enabled virtual network sharing and creation, both on-premises and on Azure, along with dedicated ExpressRoute connections between on-premises and Azure datacenters. By managing our virtual network infrastructure using a telecommunications subscriber model, we made significant network logistics and traffics changes without having to change the underlying network hardware. In addition, we automated the change process to eliminate the potential for human error or accidental duplication. The goal of our network virtualization was to:
• Remove the requirement for changes to the physical network.
• Allow for automated creation and population of subnet and address space information.
• Create a consistent network security environment using network security groups and subnet access control lists.

Moving forward with cloud automation

Using automation within Azure and our general cloud environment has provided us with the capability to provide a robust and consistent cloud management experience to our users. Specifically, implementing cloud automation has provided the following benefits:

• **A reduction in incidents caused by change.** Because of the consistency of our configuration processes, the Azure environment is less prone to issues that might be caused by incorrect information being used during a change procedure. This results in cleaner data and fewer problems with the cloud infrastructure environment. We experienced a 60 percent reduction in change-related incidents.

• **Automations tools force tracking the change history.** Our automation tools provide the benefit of an automatically generated history of the changes to the cloud infrastructure environment. Although manual touches may have gone undocumented or undetected in the Azure environment, our automated solution provides automatic logging, tracking, and cataloging of Azure management tasks.

• **Cost savings.** We have realized a general cost savings from using automated processes within Azure. These cost savings range from reductions in subscription costs to reduced labor hours required to perform a cloud deployment or management task. For example, here are some of the most significant areas in which we realized cost savings:

  • **Decreasing the number of changes required.** Automation and the consistency it brings has enabled us to significantly reduce the number of changes required in the environment. Some of this stems from a more consistent and complete configuration when a resource is deployed. It also comes from not having to make changes to correct human error during a deployment or configuration task. Our old service level agreement timelines went from 2 days to 30 minutes.

  • **Vendor savings.** We employ vendors to perform many management and deployment tasks within our cloud infrastructure environment. The reduction in manual touches required by our automation solution has reduced the vendor costs associated with those manual touches.

We are continuing to improve our tooling and user interfaces for Azure automation. We are also incorporating new Azure features and functionality as they become available on the platform. We expect even greater returns in efficiency and cost savings after more of our environment transfers over to the automated toolset.

For more information

Microsoft IT Showcase

http://www.Microsoft.com/itshowcase

© 2016 Microsoft Corporation. All rights reserved. Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.