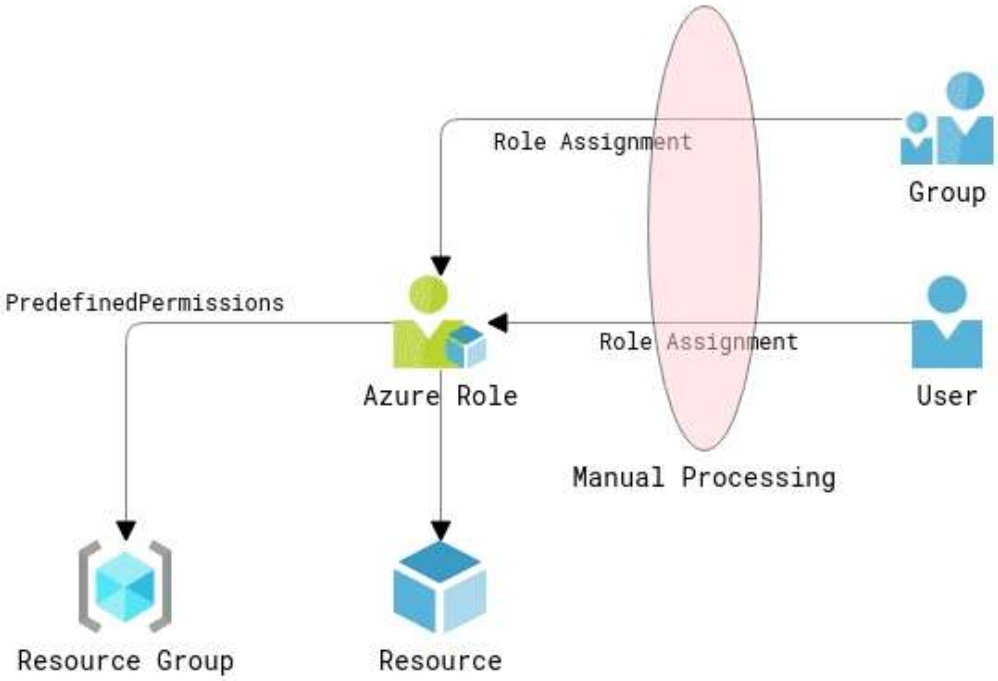
18 JANUARY 2021 / MICROSOFT AZURE

Design the access management process for Azure resources

Access management in Azure is a broad and complex topic consisting of many interconnected parts, including Azure Active Directory, the Role-based Access Control (RBAC) model, resource permissions, service-specific access configurations, and so on. Here, I would like to give you a few examples of how you can improve resource access management in Azure as a part of your Cloud Governance strategy.

The inputs



Regardless of organization size or the number of deployed Azure resources, I regularly encounter the following common scenario:

An organization has multiple Azure resources in operation, and it provides access to different teams and individuals to manage those resources according to company goals. Business as usual, as you might say. However, the way that access is provided is often quite far from operational ideals. Permissions are granted by creating Azure Role assignments for both user and group accounts. The assignments are created manually at the resource group and resource levels. No unified access model is in place, and the user experience is inconsistent. At best, access requests are submitted via an ITSM system and processed centrally by IT operations, creating historical records for security audits. At worst, users just granted Owner permissions on resource groups and manage access in whatever way they like.

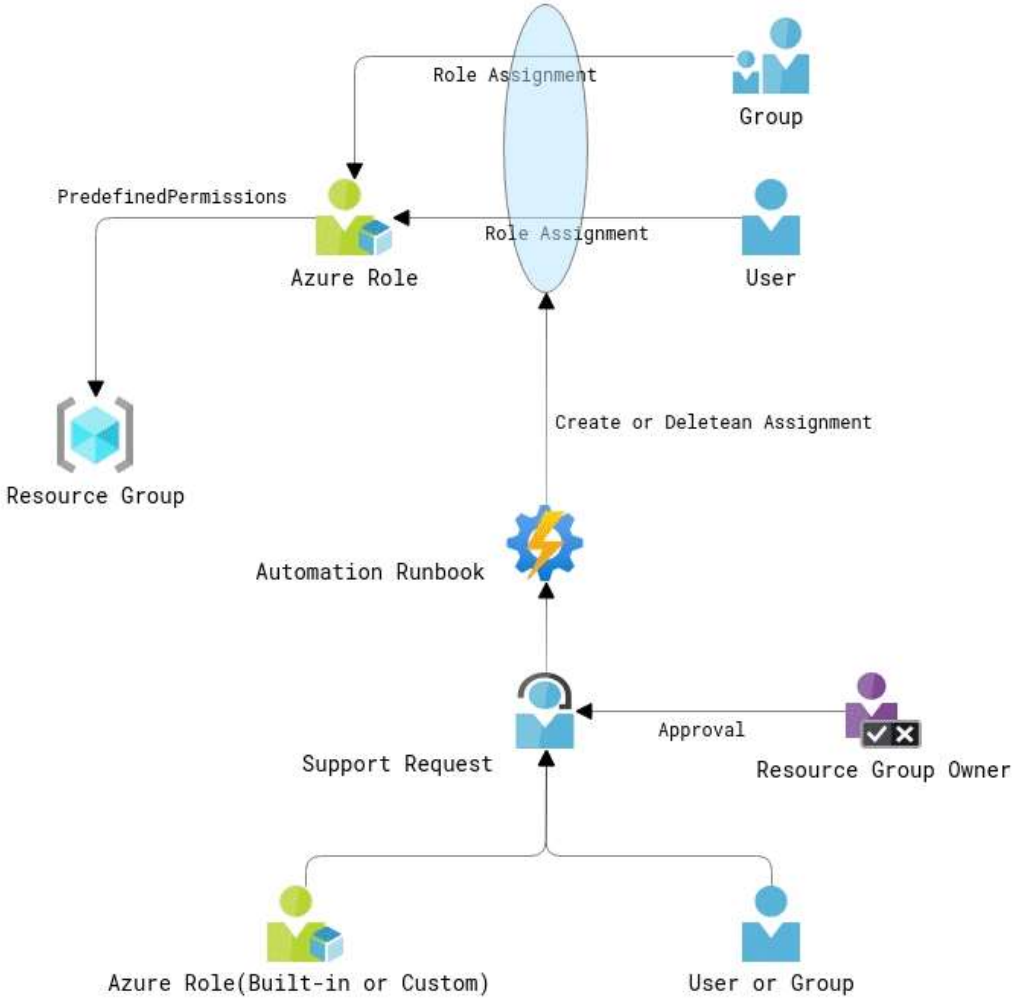
> *Azure built-in Owner and User Access Administrator roles allow their members to manage access and assign roles in*

> *Azure RBAC.*

Apart from the apparent disadvantages of the described approach like manual work, inconsistency, and delays in getting access, it contains many hidden pitfalls. For instance, you can quickly exceed the limit of 2000 role assignments per subscription and won't be able to grant new accesses. The users are likely to be granted insufficient or excessive permissions. In the latter case, overprivileged users with owner permissions might well create or edit role assignments on their own, bypassing the regular access management routine.

So, can we design a solution that does not require granting application owners permissions to manage resource access by uncontrollably creating new Azure role assignments and being user-friendly and efficient at the same time?

Option #1: A minimum investment



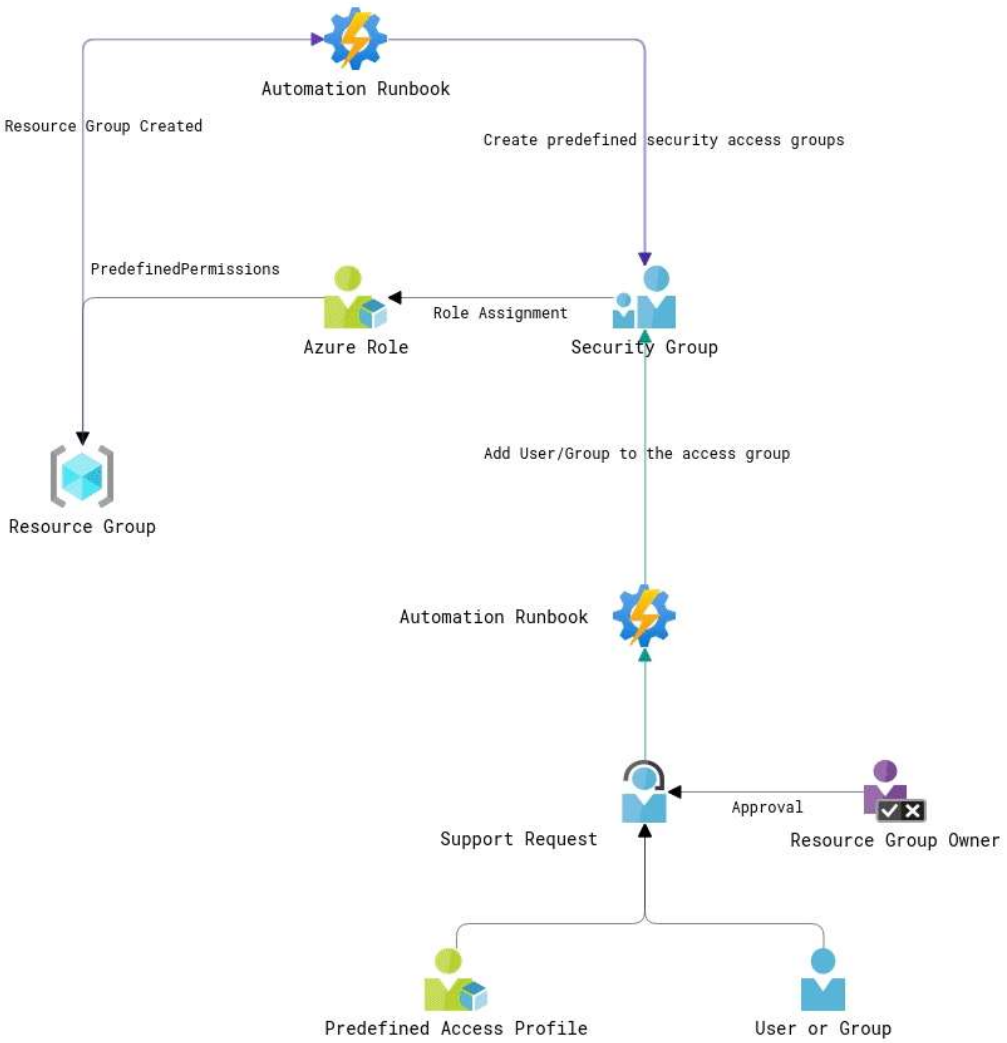
Let's start with minimalistic improvements to the initial design. As you might notice from the diagram above, the two major changes in the access management process are granting access at the resource group level only and automation of role assignment provisioning.

Firstly, granting access at the resource group level will help to simplify your access control lists (ACL) and reduce the number of role assignments. Sure, you won't get the same granular access level as with creating role assignments at the resource level. Still, it will be a reasonable tradeoff in most cases, especially when you design your resource groups according to the resource lifecycle principle creating independent application services.

Secondly, the automation of role assignment creation will basically allow you to save the time you spent on doing it manually. It can be implemented as an Azure Automation runbook, a Logic App, an Azure Function, or by any other technical means that can invoke Azure REST APIs. The automation app will require a resource group, an Azure role and user or group identity as inputs. I suggest programming the create, update and delete (CRUD) operations as separate functions to keep your automation logic clean and organized.

Finally, the process entry point can be defined as a request in your ITSM solution. To get access, users will have to fill in the request form and submit the ticket. A resource group or application owner will have to approve the request before the ITSM triggers the automation.

Option #2: Tightening the ropes



This one is an evolution of Option # 1. The key difference with the predecessor is that instead of dynamically creating new role assignments, we will be adding users to the security groups with predefined permissions. That will come to the rescue in large environments where Azure subscriptions contain hundreds of resource groups, and you might hit the limit of 2000 role assignments per subscription quite fast. Apart from that, all resource access will be managed via the security groups, which security administrators usually prefer.

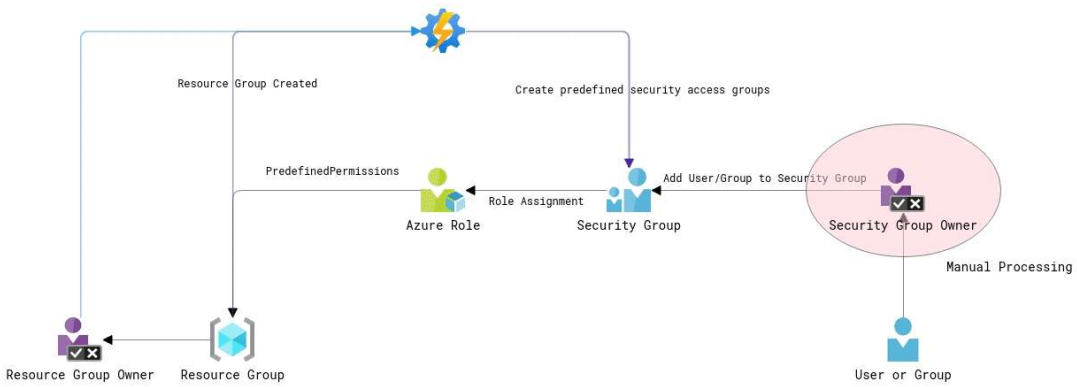
That process design uses two automated steps. The first automation is triggered when a new resource group is created (for instance, you can use an Activity log alert for that). It will generate

several security groups and provision role assignments for them using predefined Azure roles (built-in or custom). Those security groups can be either cloud-based or synced from your on-premise AD domain. The second automation adds/deletes users from those access groups. The process initiation and access approval steps remain the same.

Of course, such a design has its drawbacks. First of all, it's more complicated and requires more effort to implement compared to Option # 1. Secondly, the provisioning of predefined access groups for your Azure resource groups creates a bunch of security groups that might never be used. Besides, if you use on-premise groups for access provisioning, you should count for the synchronization delays between your AD controller and Azure Active Directory.

For sure, you can discard creating the access groups upon resource group creation and come up with automation that will create them on demand when there is an access request created. However, that approach will require creating more complex automation logic: verifying for existing access groups, creating a new access group (if needed) as a part of access request flow, and finally adding users to that group.

Option #3: A self-service kiosk



The third option is all about self-service. It can be a good choice for non-production environments and the teams with full ownership over their applications, aka “you build it, you run it.”

The automation part is involved only when a resource group is created, and it creates several predefined security groups and role assignments for them, similar to Option # 2. In addition, to creating those predefined groups, the automation also sets their owner who is the actual resource group (or application) owner approving access to the resources. By doing so, the application owner gets the ability to manage the membership of those security groups.

When users need access to resources in a specific Azure resource group, they should reach out to the resource group owner about that. The resource group owner has enough permission to add the user to the security group granting sufficient access to the resources without involving any third party. In many cases, the resource group owner and the person requesting access work on the same team, thus making the whole process much faster and efficient as those teammates don’t have to submit support tickets and rely on other teams to fulfill their needs.

As the resource group owner technically doesn’t have Owner permissions over the resource group, they cannot modify role assignments on the resource group directly. He or she will have to use the predefined security groups to manage access. That will keep the structure and the number of role assignments in a subscription organized and auditable.

Final notes

Careful readers might notice that I omitted some aspects of the described designs. Indeed, for the sake of simplicity, I didn’t go

into details about the required internals of ITSM solutions and getting the information about resource/application owners as that can be managed in different ways.
It is highly likely that in different ITSM systems, the approvals will have some internal configuration nuances that will depend on your system and its configuration.

If you store the information about resource owners in Azure tags, as I described in my posts about running CMDB for Azure, you can quite easily utilize it in your automation scripts. If that information is stored somewhere else, you will need to find your way to get it.

I hope that those sample process designs will provide us with some ideas of how you can improve access management for Azure resources in your environments. Put your question and thoughts about it into the comments, if you would like to know more about that topic 👇

— Andrew's blog —

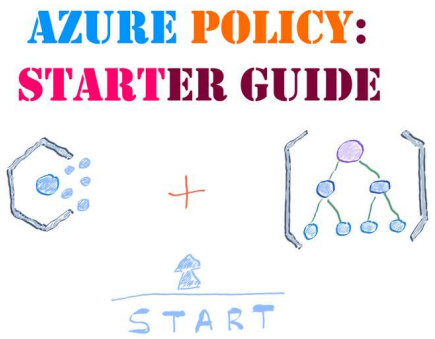
Microsoft Azure

A one-click Ghost deployment on Azure Web App for Containers

Practical aspects of running a CMDB for Azure resources: Tips

Practical aspects of running a CMDB for Azure resources: Fundamentals

See all 16 posts →



AZURE POLICY

Azure Policy: Starter Guide

My coworkers and teammates often reach out to me with similar questions regarding the best practices for creating and applying Azure Policy. That tendency encouraged me to compile this starter guide for Azure Policy, which is based on my practical experience in multiple projects.

8 MIN READ

MICROSOFT AZURE

A one-click Ghost deployment on Azure Web App for Containers

Even though I'm pretty satisfied with running my blog on a DigitalOcean droplet backed up with Cloudflare CDN, I spent some time last few months looking for an option to run Ghost on Azure.

9 MIN READ