

Reducing risk for your Azure resources

Audun Solemdal

solom

A solid orange bar with a slight gradient, darker on the left and lighter on the right, spanning the width of the slide at the bottom.

About

- Audun Solemdal
- CEO @ **solom**
- Cloud Consulting
- Azure-based Platform
- Whatever needs improvement...

 solom.no

 [solomno](#) / [audunsolemdal](#)

 [audun-solemdal](#)

Agenda

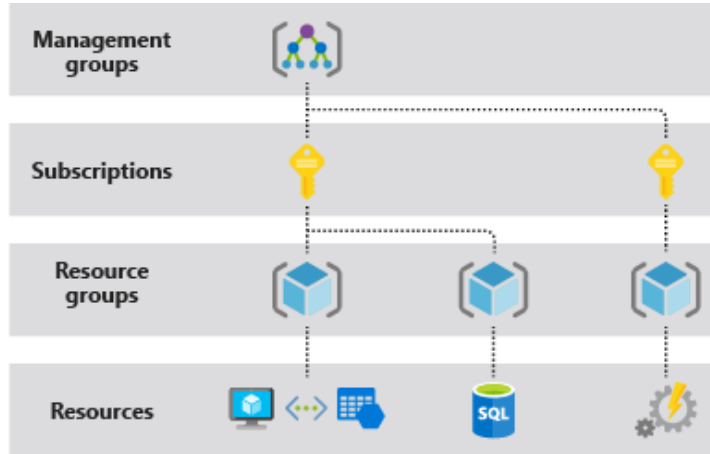
- Common RBAC assignments and their issues
 - Quick recap of Azure RBAC
 - Common RBAC handling
 - How to get your developers on board
 - Practical implementation of least-privilege
- New way to reduce scope for Owner / User Access Administrator roles
- Blocking resource deletes with DenyAction Azure policies (if time)

Quick RBAC recap

- Access is built on ARM which uses resource providers
 - E.g. Microsoft.Web, Microsoft.Storage, Microsoft.Compute
- Control plane & data plane
 - Actions, NotActions – control plane
 - DataActions, NotDataActions – data actions
- Role assignments can grant RBAC roles to principals
 - Examples - Entra ID groups, users, managed identities etc.

Quick RBAC recap

- Role assignments can be granted at different scopes and are inherited in a hierarchy
 - Up to recently very challenging to block inheritance*



* Deployment stacks (preview) may finally make this more manageable

Common RBAC handling

```
// This can also apply at resource group level depending on your governance model
private void ReduceStressLevel(string devTeam, string? leadDeveloper)
{
    if (devTeam.IsCryingAboutAzurePermissions())
    {
        try
        {
            return GrantSubscriptionAccess(devTeam, "Contributor");
        }
        catch (StillCryingAboutAzurePermissionsException)
        {
            return GrantSubscriptionAccess(leadDeveloper, "Owner");
        }
    }
}
```

Common RBAC handling

- Contributor role
 - Used because it is a practical «catch-most» issue handling
 - Grants permissions to the control plane **only**
 - Some unfortunate accesses most principals shouldn't have
 - PaaS – often involves access to admin credentials
 - IaaS – reset admin password / SSH key / run command
 - CRUD to any resource. Is this really required if you use IaC?

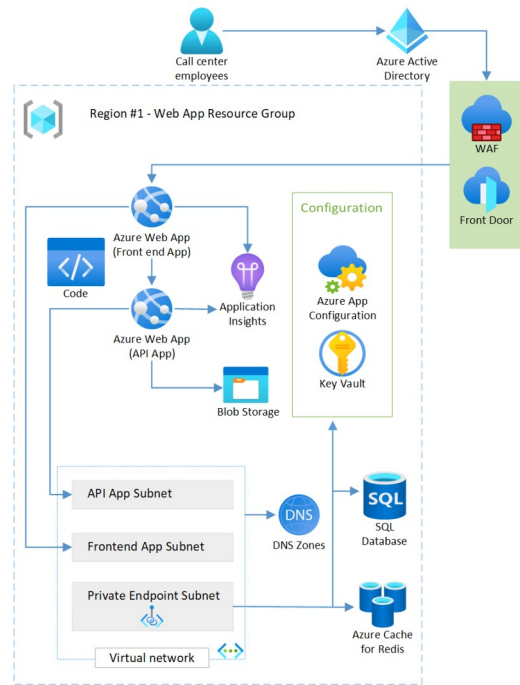
Common RBAC handling

- Owner role
 - Any action in the control plane possible, no data plane access!
 - Essentially everything a contributor can do + more
 - Most relevant: Microsoft.Authorization resource provider
 - Write and delete role assignments to anyone in your directory
 - Create and delete resource locks
 - Assign, modify and delete Azure Policy assignments

Authenticating with the data plane

- Nearly all Azure PaaS services supports using keys for authentication
 - Contributors have access to these keys. Since developers are often granted contributor permissions, these keys are often used in app code (with or without Azure Key Vault)
- Keys should ideally be removed or disabled wherever possible
 - Assign RBAC roles with the required data plane permissions instead.
 - For greenfield development – disable all keys for PaaS services by default

Access keys and Action roles



- How many keys or «Action-centric» role assignments are required for your managed identities and developers?
- Including preview functionality – 0 Action-centric roles, 1 or 0 keys (app insights connection string from frontend)

Least privilege for the API

- Azure RBAC roles
 - Storage Blob Data Contributor / Reader
 - Monitoring Metrics Publisher (preview in SDK versions)
 - Key Vault Secrets User
 - App Configuration Data Reader
- Service-specific roles
 - SQL database
 - "CREATE USER [my-app-name] FROM EXTERNAL PROVIDER; ALTER ROLE(..)»
 - Also possible to grant Entra admin at server level if feeling frisky..
 - Redis cache - (preview) «Data Contributor» or custom access policy

DefaultAzureCredential

- Typically you only need to change this part of your app code
 - This exact example requires .NET 6 or higher

```
// Before
using Azure.Messaging.ServiceBus;

var client = new ServiceBusClient(connectionStringWithSecret);
client.CreateSender(queueName);

// After
using Azure.Messaging.ServiceBus;
using Azure.Identity;

var client = new ServiceBusClient(connectionStringWithoutSecret, new DefaultAzureCredential());
client.CreateSender(queueName);
```

- DefaultAzureCredential
 - ManagedIdentityCredential
 - WorkloadIdentityCredential
 - AzureCliCredential
- Workload Identity
 - Kubernetes, Github, Azure DevOps ++
- Same principle can work against other Microsoft services
 - Microsoft Graph

Azure DevOps service connections...

- Check if your Sub / RG IAM looks something like this
 - This can now easily be improved via a few clicks
 - Also consider manual assignment to reduce the permissions

App	Contributor ⓘ	Subscription (Inherited)
App	Contributor ⓘ	Subscription (Inherited)
App	Contributor ⓘ	Subscription (Inherited)
App	Contributor ⓘ	Subscription (Inherited)
App	Contributor ⓘ	Subscription (Inherited)
App	Contributor ⓘ	Subscription (Inherited)

Practical implementation in one slide

- Strongly consider IaC + Git to make this manageable!
- Grant needed roles access to groups and managed identities only
- Inform your developers how to deal with this
- Disable all keys / passwords wherever possible
- Suggested role assignments
 - Entra ID groups with users
 - create two baseline «platform level» custom roles, assign to RG/Sub
 - Or Assign every role assignment needed to the group
 - In production – consider PIM for Azure resources / PIM for groups
 - Managed identity / Entra ID group with identities
 - Assign every role assignment needed to the group

Questions regarding part 1?

New ABAC possibilities

- Reducing scope of Owner / User Access Administrator roles
- The problem with delegating user access is limiting which principals and which roles can be assigned
- In preview – great improvements on limiting this
- Working Terraform code
 - <https://github.com/solomno/sharing/tree/main/terraform/rbac>
- Quick demo..

Questions regarding ABAC?

DenyAction Azure policies

- Up to this point we have looked at reducing permissions
- In the end, some admins will still need privileged access to resources
- How to deal with this

DenyAction Azure policies

- Resource locks
 - Inheritance based
 - Can be bothersome on nested objects
 - Only works on resources which support location and tags
 - Locks can be removed by Owners / User Access Administrators
 - Protection against resource moves(?)

DenyAction Azure policies

- DenyAction effect Azure policies
 - Supported on non-indexed objects
 - Works better with nested objects
 - Can only be removed by adjusting central policy
 - Does not care what IAM role you are assigned
 - Can be used in conjunction with resource locks to fill the gaps

DenyAction Azure policies

- Recommended
 - Protect resources which should rarely, if ever be deleted
 - Azure DNS Zones
 - Azure Firewall
 - Container Registry
 - VWAN / Hub
 - WAF
 - Public IPs at specific scopes
 - **LTR SQL backups**
 - And more..
 - Sample policy
 - <https://github.com/solomno/sharing/tree/main/terraform/policy>

Questions regarding DenyActions?

Thank you for attending!

Audun Solemdal

solom



[audun-solemdal](#)



[solomno/sharing](#)



[solom.no](#)