**Manage Azure Subscriptions and RBAC**

* Understanding Azure Subscriptions
* Configuring Role Based Access Control
* RBAC using Portal
* RBAC using PowerShell and CLI
* Custom Roles for RBAC

**Understanding Azure Subscriptions**

An Azure subscription is a logical unit of Azure services that is linked to an Azure account. Billing for Azure services is done on a per-subscription basis.

**Default Roles Assigned to User when the first subscription is created:**

1. **Global Administrator** for AD Tenant
2. **Account Administrator** is able to manage billing and invoice related issues of Subscription.
3. **Service Administrator** is to manage Azure Services like VM, Storage etc...

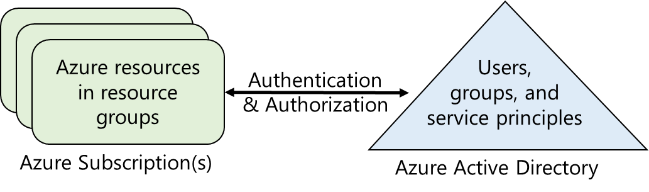
Note:

Only Account Administrator can change Service Administrator and Service Administrator has equal permissions as Owner Role of the Subscription.

From a Subscription, owner can be removed, Service Administrator cannot be removed.

**Access control in Azure starts from a billing perspective.**

* The actual owner of an Azure account is the Account Administrator (AA).
* **Subscriptions are a container for billing**, but they also act as a security boundary.
* **Your Azure subscription has a trust relationship with Azure AD**, which means that it trusts the directory to authenticate users, services, and devices.
* Multiple subscriptions can trust the same directory, but each subscription trusts only one directory.

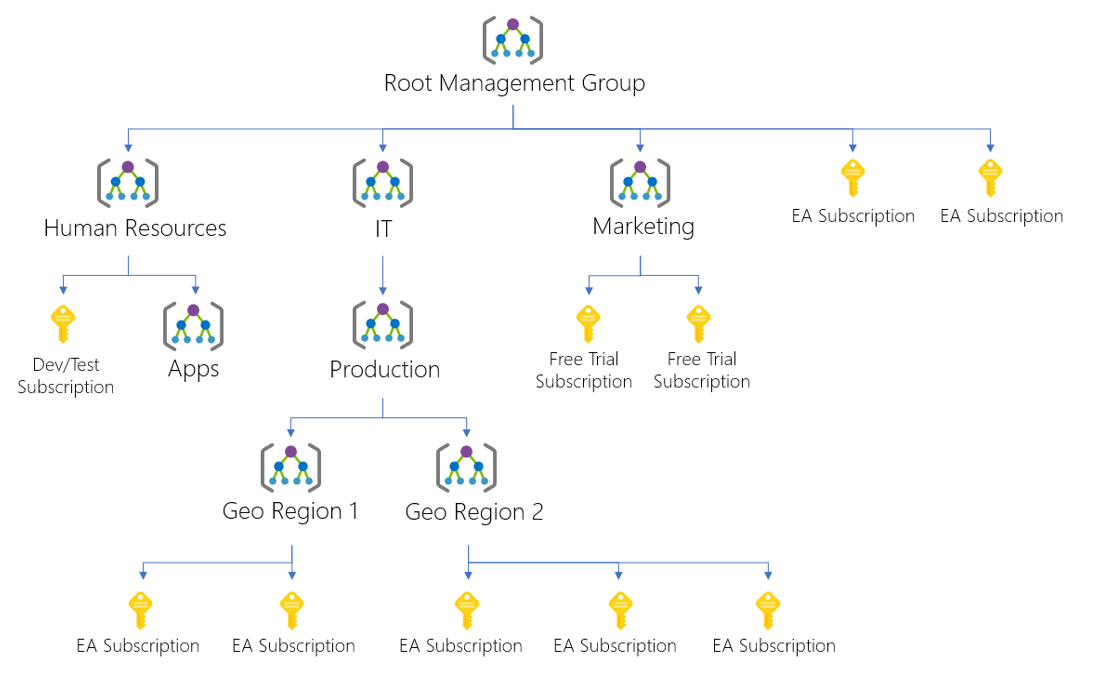


* Permission Scopes (Users can be given access to)
  + Management Group
    - Management Group(s)
    - Subscription(s)
      * ResourceGroup(s)
        + Resource(s)

**Management Group**

**About Management Group**

* Provides a level of scope above subscriptions.
* Targeting of policies and spend budgets across subscriptions and inheritance down the hierarchies.
* We can have upto 6 levels of management group.

****

**About Resource Group**

* A resource groups is a fundamental concept of the Azure platform.
  + Serves as a logical grouping of resources
  + Ties to resource life cycle
  + Can't be nested
* Each resource must belong to a resource group.
* Most resources can be moved between resource groups.
* A resource in a resourse group is not required to have same region/location as resource group.
* Organization of Resource Group
  + Organizing for authorization
  + Organizing for life cycle
  + Organizing for billing

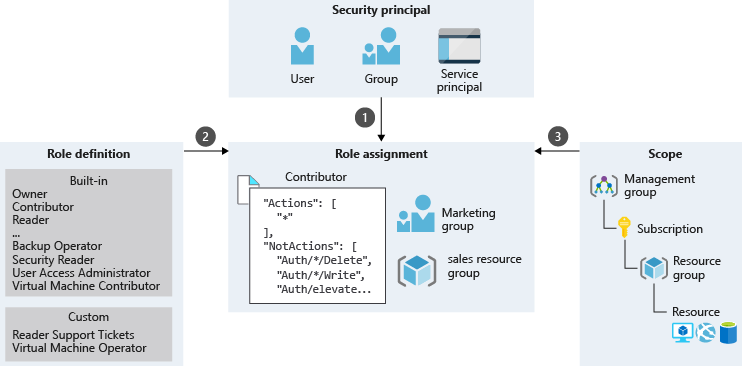
Graphical user interface

Description automatically generated

**Configuring Role Based Access Control (RBAC)**

* Managing access to resources in Azure is a critical part of an organization’s security and compliance requirements. Role-based access control (RBAC) is the capability for you to grant appropriate access to Azure AD users, groups, and services.
* RBAC is configured by selecting a role (the definition of what actions are allowed and/or denied), then associating the role with a user, group or service principal.
* **Finally, this combination of role and user/group/service principal is scoped to either the entire subscription, a resource group, or specific resources within a resource group.**

**Role Assignment:**



**Role Definition (What you can do):**

Each role is a set of properties defined in a **JSON** file. This role definition includes **Name**, **Id**, and **Description**. It also includes the allowable permissions (**Actions**), denied permissions (**NotActions**), and **scope** (read access, etc.) for the role.

**Name**: Owner

**ID**: 8e3af657-a8ff-443c-a75c-2fe8c4bcb65

**IsCustom**: False

**Description**: Manage everything, including access to resources

**Actions**: {\*}

**NotActions**: {}

**DataActions**: {}

**NotDataActions**: {}

**AssignableScopes**: {/}

In this example the Owner role means all (\*) actions, no denied actions, and all (/) scopes.

{

    "id": "/providers/Microsoft.Authorization/roleDefinitions/8e3af657-a8ff-443c-a75c-2fe8c4bcb635",

    "properties": {

        "roleName": "Owner",

        "description": "Grants full access to manage all resources, including the ability to assign roles in Azure RBAC.",

        "assignableScopes": [

            "/"

        ],

        "permissions": [

            {

                "actions": [

                    "\*"

                ],

                "notActions": [],

                "dataActions": [],

                "notDataActions": []

            }

        ]

    }

}

**Actions:**

It specifies the Azure operations to which the role grants access. It is a collection of operation strings that identify securable operations of Azure resource providers.

Operation strings follow the format of **Microsoft.<ProviderName>/<ChildResourceType>/<action>**.

Examples:

* \*/read grants access to read operations for all resource types of all Azure resource providers.
* Microsoft.Compute/\* grants access to all operations for all resource types in the Microsoft.Compute resource provider.
* Microsoft.Network/\*/read grants access to read operations for all resource types in the Microsoft.Network resource provider of Azure.
* Microsoft.Compute/virtualMachines/\* grants access to all operations of virtual machines and its child resource types.
* Microsoft.Web/sites/restart grants access to restart websites.

**NotActions:**

Use the **NotActions** property if the set of operations that you wish to allow is more easily defined by **excluding** **restricted** **operations**. The access granted by a custom role is computed by subtracting the **NotActions** operations from the **Actions** operations.

**AssignableScopes:**

This property of the role specifies the scopes (subscriptions, resource groups, or resources) within which the custom role is available for assignment.

* + /
  + /subscriptions/[subscription id]
  + /subscriptions/[subscription id]/resourceGroups/[resource group name]
  + **/subscriptions/[subscription id]/resourceGroups/[resource group name]/[resource]**

**Built-in Roles and their Action and NotActions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Action** | **NotActions** | **Description** |
| **Owner** | \* | - | This role has full access to all the resources and can **delegate** access to others. |
| **Contributor** | \* | Microsoft.Authorization/\*/Delete,  ‎Microsoft.Authorization/\*/Write, | This role can **create and manage** all types of resources, but **can’t grant access** to other users and groups. |
| **Reader** | \*/read | - | This role can **view** existing Azure resources |

**RBAC supports *deny* *assignments*:**

* Attaches a set of deny actions to a user, group, service principal, or managed identity at a particular scope for the purpose of denying access.
* Deny assignments block users from performing specified actions even if a role assignment grants them access.
* Deny assignments take precedence over role assignments.
* At this time, the only way you can add your own deny assignments is by using **Azure Blueprints**.

**Custom Roles for RBAC**

**Create custom roles for Azure Role-Based Access Control**

The following template shows a custom role for **monitoring and restarting virtual machines**:

**d:\VMOperator.json**

{

"Name": "**Virtual Machine Operator**",

"Id": "97e15602-5e9d-4f79-b737-ae959719b65d",

"**IsCustom": true**,

"Description": "Can monitor and restart virtual machines.",

**"Actions":** [

"Microsoft.Storage/\*/read",

"Microsoft.Network/\*/read",

"Microsoft.Compute/\*/read",

"Microsoft.Compute/virtualMachines/start/**action**",

"Microsoft.Compute/virtualMachines/restart/**action**",

"Microsoft.Authorization/\*/read",

"Microsoft.Resources/subscriptions/resourceGroups/read",

"Microsoft.Insights/alertRules/\*",

"Microsoft.Insights/diagnosticSettings/\*",

"Microsoft.Support/\*"

],

**"NotActions":** [

],

"**AssignableScopes**": [

"/subscriptions/c276fc76-9cd4-44c9-99a7-4fd71546436e",

]

}

Then you use the [New-AzRoleDefinition](https://docs.microsoft.com/en-us/powershell/module/azurerm.resources/new-azurermroledefinition) or [az role definition create](https://docs.microsoft.com/cli/azure/role/definition#az-role-definition-create) commands to create the custom role.

New-AzRoleDefinition -InputFile VMOperator.json