Azure role-based access control (Azure RBAC)

- The policy of any organization is to follow the principles of least privileges. One must not be given access beyond what is necessary to perform a role in the organization.
- The principles apply for cloud resources also. Let's take the example of a VM operator. His role dictates that he must be able to **start/stop/restart/create/delete** VMs.
- So, we use Azure **RBAC** to grant just that access. In our case, we will grant the operator the RBAC role of VM contributor.
- Azure **RBAC** is an authorization system. It uses Azure Resource Manager behind the scenes. Azure RBAC provides fine-grained control of access to Azure resources at various levels.
- The Policies can be applied with a boundary like being able to do so in a set of resource groups called scope.

Let's see some examples where we use Azure RBAC:

- Grant access DBA group to manage databases in 2 resource groups.
- A user can manage all resources in a resource group like VM, web apps, storage account, Vnet/Subnets.
- Grant one application to access to create resources.

How Azure RBAC works

We assign Azure roles to make RBAC work. A role assignment consists of three elements: security principal, role definition, and scope.

1. Security principal

A *security principal* is an object that could represent a user or a group or a service principal or managed identity and requests access to Azure resources. We can grant access to any of these entities.

2. Role definition

A *role definition* is a collection of permissions and is called a *role*. A role definition will list the operations that can be performed. It could be something like read, write, and delete. We could grant access at a high level like owner, or even more specific roles like the VM operator where the access is limited to VM operations only.

Azure has *built-in roles* that you can use. For example, we have a contributor role where we can create all objects but we cannot grant. If we want to grant access to only certain resources, we will create a custom-defined role.

As you can see below, we can apply policies against the data stored within the scope's resources. For example, the secret within a key will be data, and we can dictate whether the data can be read or not.

ROLE DEFINITION

```
"permissions": [
    {
      "actions": [
        "Microsoft.Authorization/*/read",
        "Microsoft.Insights/alertRules/*",
        "Microsoft.Resources/deployments/*",
        "Microsoft.Resources/subscriptions/resourceGroups/read",
        "Microsoft.Support/*",
        "Microsoft.KeyVault/checkNameAvailability/read",
        "Microsoft.KeyVault/deletedVaults/read",
        "Microsoft.KeyVault/locations/*/read",
        "Microsoft.KeyVault/vaults/*/read",
        "Microsoft.KeyVault/operations/read"
      ],
      "notActions": [],
      "dataActions": [
        "Microsoft.KeyVault/vaults/*"
      ],
      "notDataActions": []
    }
  ],
  "roleName": "Key Vault Administrator",
  "roleType": "BuiltInRole",      =========> Builtin or CustomRole
  "type": "Microsoft.Authorization/roleDefinitions"
}
```

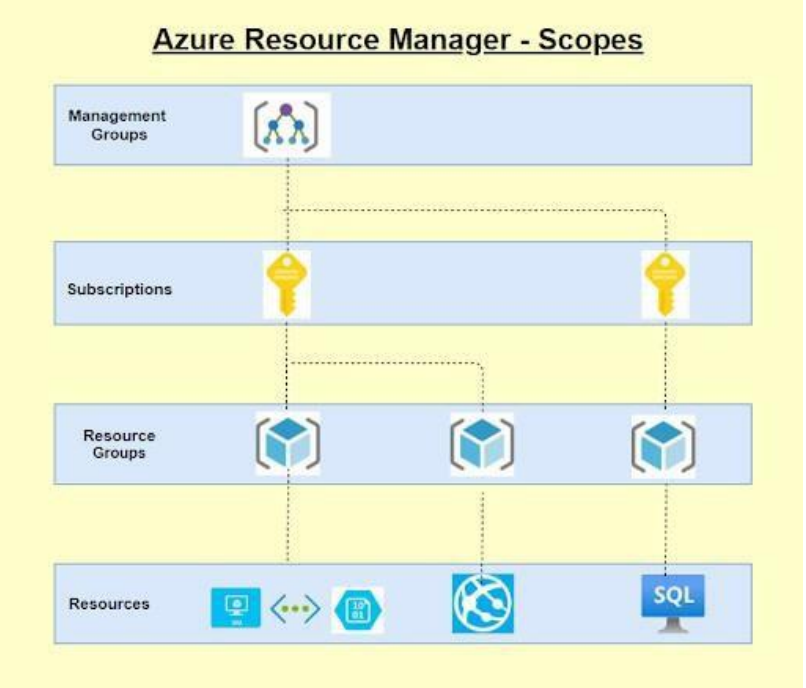
3. Scope

The scope is the set of resources to which we apply the access to. Let's say that we grant a VM operator role to a person, but we don't want that person to be able to stop VMs in production, then we apply the scope to non-production subscription or resource group only.

A scope can be applied at the four levels:

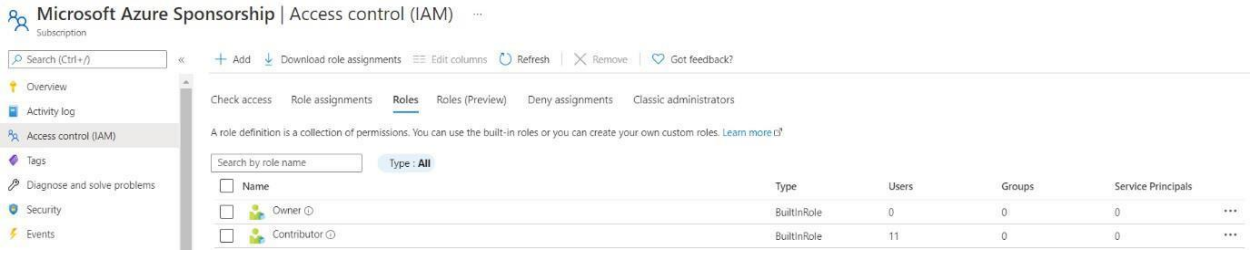
a. **Management group**
b. **Subscription**
c. **Resource group**
d. **Resource**

- Scopes follow a hierarchical structure, and they follow a parent-child relationship.
- Scopes applied at a higher level are inherited by the resources below it.
- For example, a policy with a scope of Management groups will be inherited by all subscriptions under it.
- Likewise, a policy scoped at the RG level will be inherited by all resources under it.



4. Role assignments

We assign the role to the user or group. When we assign the role, the user gets the privileges. And we simply remove the role assignment when we want to revoke the access. Under IAM, for every resource, we can see the roles under the roles tab.



5. Deny assignments

Earlier RBAC had only allowed, but now it can be denied assignments also. If there is a deny assignment, the user will be blocked from doing the action. Deny assignments take precedence over role assignments where a given user has both allow and deny but deny will be the end action.

6. License requirements

RBAC feature is free and included with our Azure subscription.