



AZ-203.4

Module 02: Implementing access control

Subtitle or speaker name



Topics

- Claims-based authorization.
- Role-based access control (RBAC) authorization.
- Virtual machine access control.

Lesson 01: Claims-based authorization



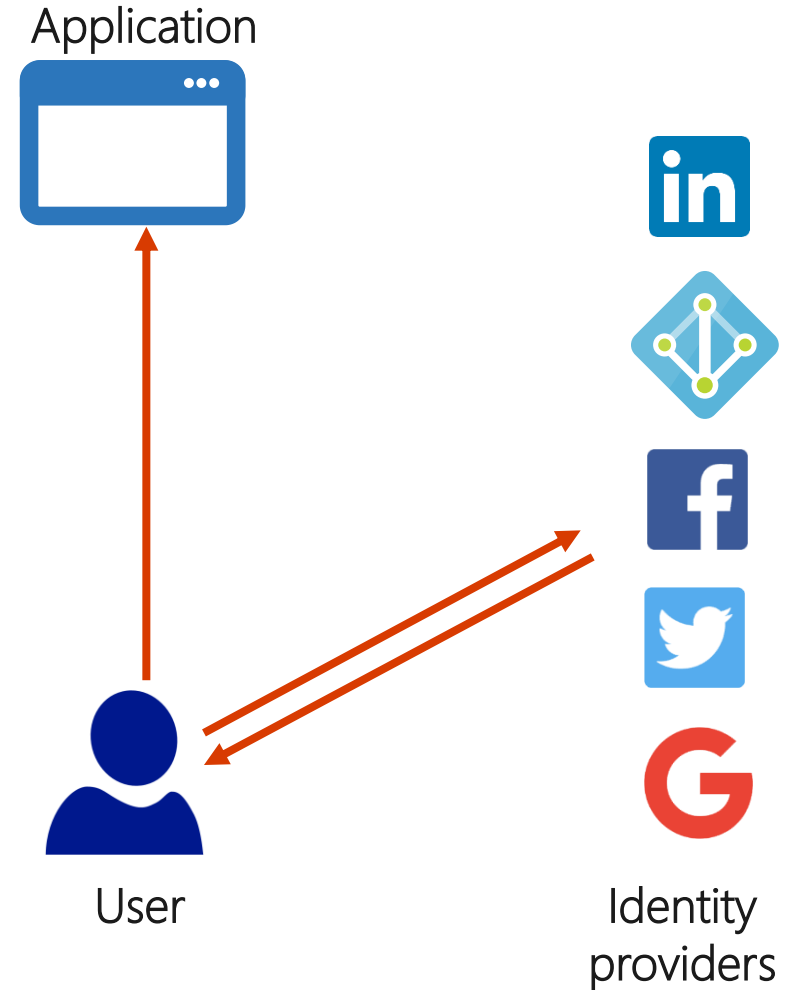
Claims: Authorization

- Process of determining which entities have permission to change, view, or otherwise access a computer resource
- Example: in a corporate application, only managers can delete a customer record
 - First, we authenticate the user
 - After authenticated, the user will request to delete a customer record
 - If the user is a manager, we then authorize the user to delete a customer record

Claims: Identity providers

Identity comes from third-party providers more often than from internal directories:

- Facebook
- Google
- LinkedIn
- Twitter




Claims: Claim example – driver’s license

ANYWHERE

NO. 123456

CLASS: D



Yarborough, Rocco M.

123 STREET

CITY, ST. 12345

DOB: 06/08/70

Height: 6’0”

Eye Color: Brown

EXP 06/08/2025

Claim	Value
Name	Rocco M. Yarborough
Date Of Birth	June 8, 1970
Height	6 feet, 0 inches
Eye Color	Brown
Authority	Contoso State Motor Vehicle Agency

Claims: Claim example – driver's license (continued)

- The driver's license payload includes four claims
 - The driver's name is Rocco M. Yarborough
 - The driver was born
 - June 8, 1970
 - The driver is 6 feet, 0 inches tall
 - The driver's eyes are brown
- The claim was issued by the Contoso Motor Vehicle Agency
 - Trust but verify our authorities
 - We trust any information given to us by this agency
 - We will verify that the claims did in fact come from this agency
 - Because we trust this agency, we can trust that the values of the claim are true

Claims-based authorization

- Instead of users belonging to specific groups, our claims will describe the user in more detail
- We use the claims to determine if the user is authorized to perform a specific action
- Example
 - User A attempts to go to the secure section of our website
 - User A is redirected to a list of identity providers
 - User A signs in by using a LinkedIn identity
 - User A gives LinkedIn permission to share data (claims) with our application
 - User A is redirected to our website with a set of claims including name and email

Claims-based authorization: Workflow

- Authentication

- User A attempts to go to the secure section of our website
- User A is redirected to a list of identity providers
- User A signs in using a LinkedIn identity
- User A gives LinkedIn permission to share data (claims) with our application
- User A is redirected to our website with a set of claims including name and email

- Authorization

- Our application reads User A's claims to determine if the user is authorized to perform certain actions
- Example: we might have a list of user emails that are allowed administrative access

Claims-based authorization: claim policy in ASP.NET

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();

    services.AddAuthorization(options =>
    {
        options.AddPolicy(
            "EmployeeOnly",
            policy => policy.RequireClaim("EmployeeNumber")
        );
    });
}
```



Claims-based authorization: enforcing claims in ASP.NET

```
[Authorize(Policy = "EmployeeOnly")]  
public class VacationController : Controller  
{  
    public ActionResult VacationBalance()  
    {  
    }  
    [AllowAnonymous]  
    public ActionResult VacationPolicy()  
    {  
    }  
}
```



Specific claim values using policies in ASP.NET

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();

    services.AddAuthorization(options =>
    {
        options.AddPolicy(
            "SpecificEmployeesOnly",
            policy => policy.RequireClaim("EmployeeId", "1", "2", "3", "4", "5")
        );
    });
}
```



Lesson 02: Role-based access control (RBAC) authorization



Role-based access control overview

- User permissions are managed and enforced by an application based on user roles
 - General access is assumed to be denied
 - If a user has a role that is required to perform an action, access is then granted
- Identities can belong to one or more roles
 - Example
 - **Holly** works for Contoso, Ltd., so she belongs to the **User** role
 - **Holly** was recently promoted to oversee a developer team, so she now also belongs to the **TeamAdmin** role
 - **Holly** has access to any operations that require the **User** or **TeamAdmin** roles

Role-based access control overview: Azure RBAC

- Provides fine-grained access management of resources in Azure
 - Built on Azure Resource Manager
 - Segregate duties within your team
 - Grant only the amount of access to users that they need to perform their jobs
- Concepts
 - **Security principal:** Object that represents something that is requesting access to resources
 - Examples: user, group, service principal, managed identity
 - **Role definition:** Collection of permissions that lists the operations that can be performed
 - Examples: Reader, Contributor, Owner, User Access Administrator
 - **Scope:** Boundary for the level of access that is requested
 - Examples: management group, subscription, resource group, resource
 - **Assignment:** Attaching a role definition to a security principal at a particular scope
 - Users can grant access described in a role definition by creating an assignment
 - Deny assignments are currently read-only and can only be set by Azure

Role definition

Owner
Contributor
Reader
...
Backup Operator
Security Reader
User Access Administrator
Virtual Machine Contributor

Built-in

Reader Support Tickets
Virtual Machine Operator

Custom

Contributor

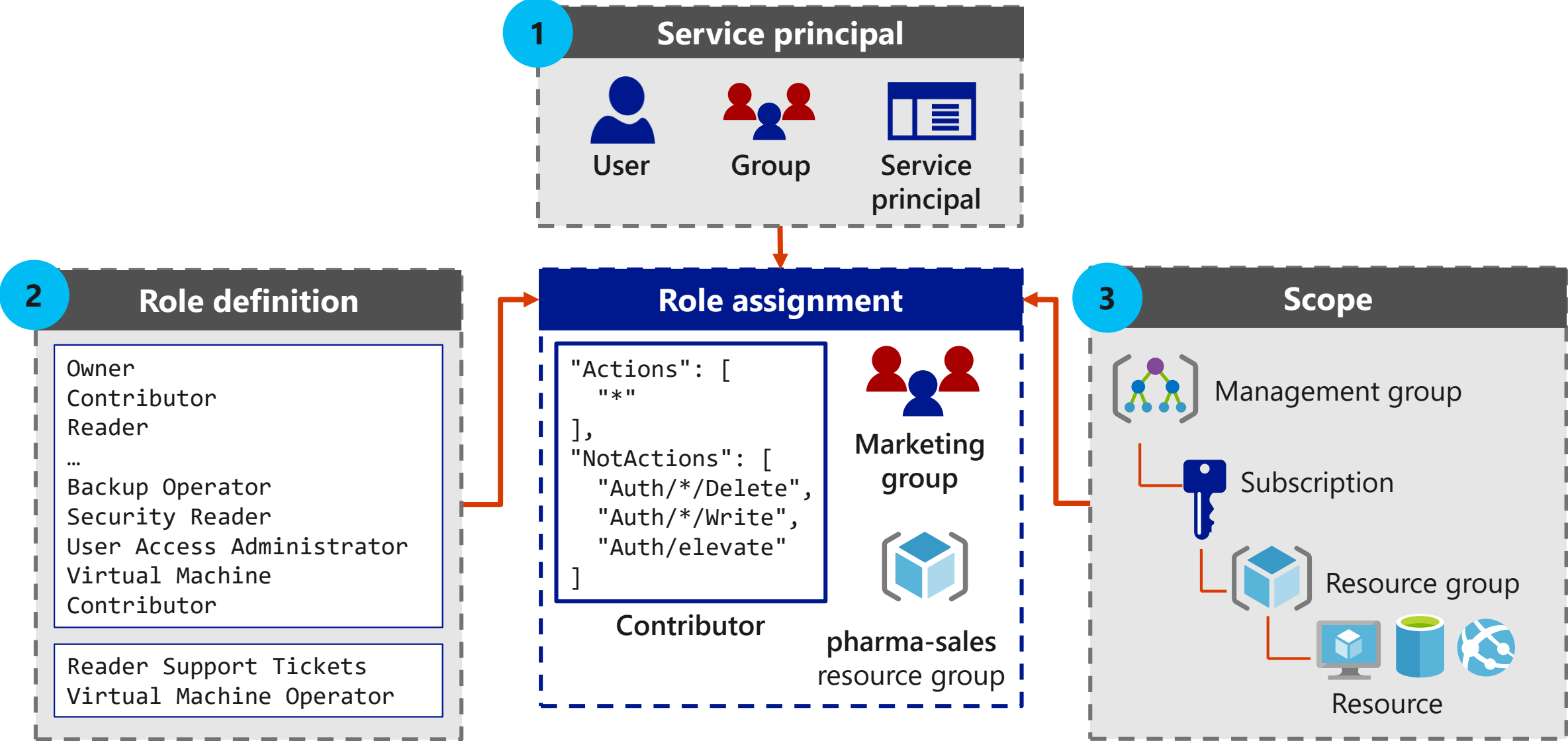
```
"Actions": [  
  "*"   
],  
"NotActions" : [  
  "Authorization/*/Delete",  
  "Authorization/*/Write",  
  "Authorization/elevateAccess/Action"   
],  
"DataActions" : [],  
  "NotDataActions": [],  
  "AssignableScopes" : [  
    "/"   
  ]
```


Role assignment

Process of binding a role definition to a user, group, or service principal at a particular scope for the purpose of granting access

- Access is granted by creating a role assignment
- Access is revoked by removing a role assignment

Role assignment (continued)



Demo: Observe RBAC assignments



Role-based access control overview: Azure RBAC (continued)

How RBAC determines if a user has access to a resource:

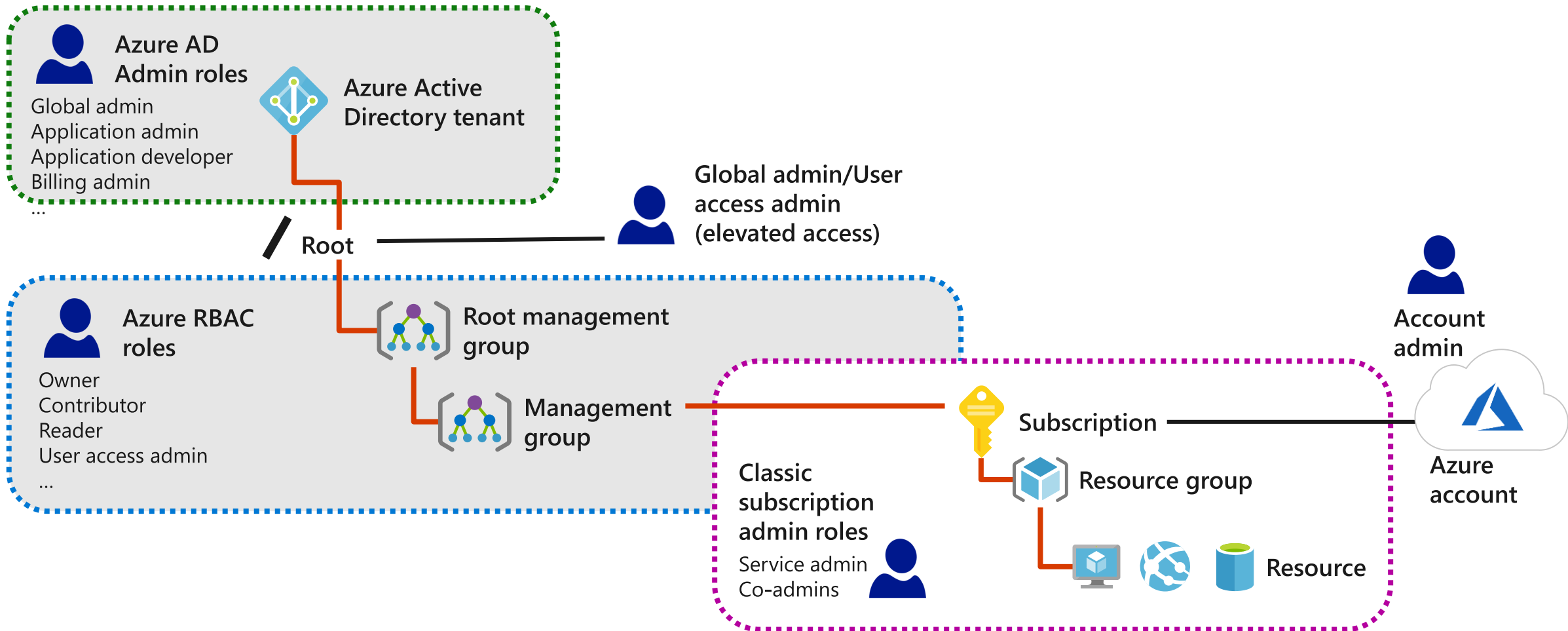
1. A user (or service principal) acquires a token for Azure Resource Manager.
2. The user makes a REST API call to Azure Resource Manager with the token attached.
3. Azure Resource Manager retrieves all the role assignments and deny assignments that apply to the resource upon which the action is being taken.
4. Azure Resource Manager narrows the role assignments that apply to this user or their group and determines what roles the user has for this resource.
5. Azure Resource Manager determines if the action in the API call is included in the roles that the user has for this resource.
6. If the user doesn't have a role with the action at the requested scope, access is not granted. Otherwise, Azure Resource Manager checks if a deny assignment applies.
7. If a deny assignment applies, access is blocked. Otherwise, access is granted.

Classic subscription administrator roles, Azure RBAC roles, and Azure AD administrator roles

Azure and Azure AD offer three types of role-based access control roles:

- Classic subscription administrator roles
- Azure role-based access control (RBAC) roles
- Azure AD administrator roles

Classic subscription administrator roles, Azure RBAC roles, and Azure AD administrator roles (continued)



Azure RBAC roles

RBAC role in Azure	Permissions	Notes
Owner	Has full access to all resources and can delegate access to others	The Service Administrator and Co-Administrators are assigned the Owner role at the subscription scope. This applies to all resource types.
Contributor	Creates and manages all types of Azure resources but cannot grant access to others	This applies to all resource types.
Reader	Views Azure resources	This applies to all resource types.
User Access Administrator	Manages user access to Azure resources	This applies to managing access, rather than to managing resources.

Azure RBAC roles vs. Azure AD administrator roles

Azure RBAC roles	Azure AD administrator roles
Manage access to Azure resources	Manage access to Azure AD objects
Support custom roles	Does not support custom roles
Scope can be specified at multiple levels	Scope is at the tenant level
Role information can be accessed in the Azure portal, Azure CLI, Azure PowerShell, Azure Resource Manager templates, REST API	Role information can be accessed in Azure portal, Office 365 admin portal, Microsoft Graph, Azure Active Directory PowerShell for Graph

Manage access by using RBAC and the REST API

List access

GET `https://management.azure.com/{scope}/providers/Microsoft.Authorization/roleAssignments?api-version=2015-07-01&$filter={filter}`

Grant access

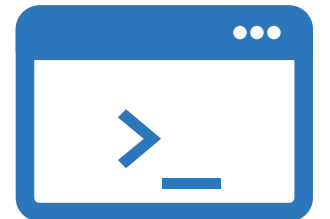
PUT

`https://management.azure.com/{scope}/providers/Microsoft.Authorization/roleAssignments/{roleAssignmentName}?api-version=2015-07-01`

```
{
  "properties": {
    "roleDefinitionId":
"/subscriptions/{subscriptionId}/providers/Microsoft.Authorization/roleDefinitions/{roleDefinitionId}",
    "principalId": "{principalId}"
  }
}
```

Remove access

DELETE `https://management.azure.com/{scope}/providers/Microsoft.Authorization/roleAssignments/{roleAssignmentName}?api-version=2015-07-01`



Role-based authorization in ASP.NET

```
[Authorize(Roles = "Administrator, PowerUser")]
public class ClockInController : Controller
{
    public ActionResult SetTime()
    {
    }

    [Authorize(Roles = "Administrator")]
    public ActionResult ShutDown()
    {
    }
}
```



Enforcing roles by using policies in ASP.NET

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();

    services.AddAuthorization(options =>
    {
        options.AddPolicy(
            "RequireAdministratorRole",
            policy => policy.RequireRole("Administrator", "PowerUser")
        );
    });
}

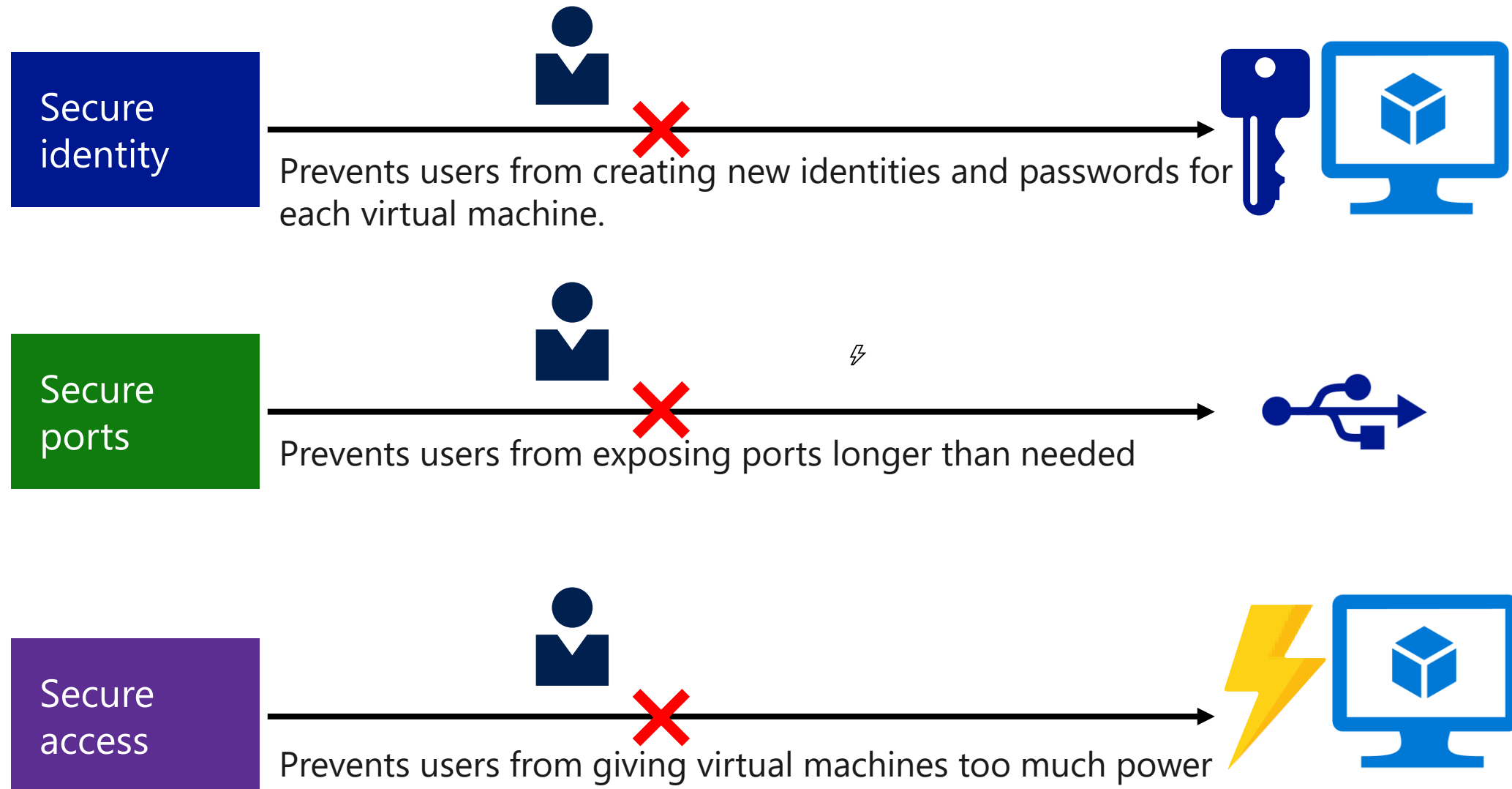
[Authorize(Policy = "RequireAdministratorRole")]
public IActionResult Shutdown()
{
    return View();
}
```



Lesson 03: Virtual machine access control



Virtual machine security exposure

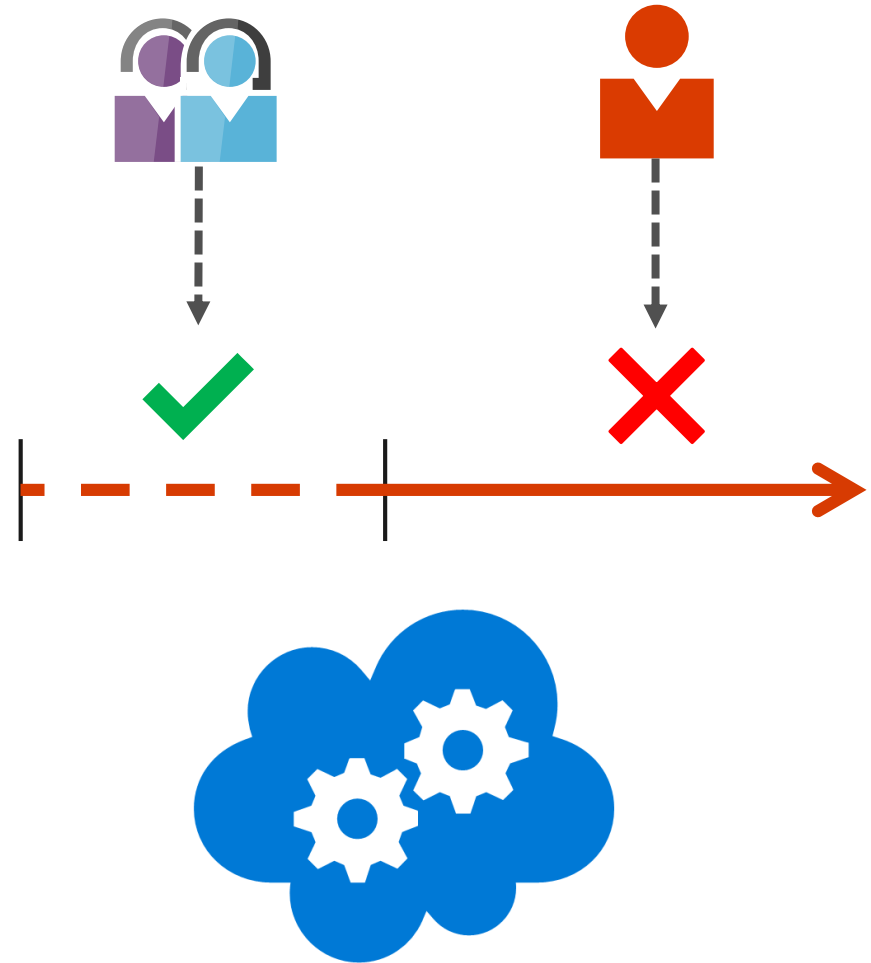


Virtual machine authentication by using Azure AD

- Use Azure AD credentials to sign in to Azure virtual machines:
 - No need to create local admin accounts
- Inherits some of the advantages of using Azure AD:
 - Password complexity and lifetime requirements
 - Multi-factor authentication
 - Federation
- Integrates with role-based access control (RBAC):
 - Manage access by using RBAC instead of manually in each VM

Just-in-time access

- Enables access when requested:
 - Access only enabled to specific ports
 - Access only enabled within time window
- Integrated with VM Connect blade in the Azure Portal



Managed identities

- Special type of service principal to be used only with Azure resources:
 - Managed by Azure AD
 - Assigned directly to an Azure resource
- Supports authentication to Azure resources from your applications without storing credentials in your code

Managed identity assignment

- System-assigned:
 - Enable directly on the Azure resource
 - Azure creates and manages the identity
 - Automatically deleted when the resource is deleted
- User-assigned:
 - Standalone resource
 - Can be assigned to one or more Azure resources
 - Separate lifecycle from assigned Azure resources

Managed identities for a VM by using Azure PowerShell

Provide an Azure AD identity for the VM

```
New-AzUserAssignedIdentity -ResourceGroupName myResourceGroupVM -Name ID1
```

Get the VM's properties

```
$vm = Get-AzVM -ResourceGroupName myResourceGroup -Name myVM
```

Assign the identity to the VM

```
Update-AzVM -ResourceGroupName TestRG -VM $vm -IdentityType "UserAssigned" -IdentityID  
"/subscriptions/<SUBSCRIPTIONID>/resourcegroups/myResourceGroupVM/providers/Microsoft.M  
anagedIdentity/userAssignedIdentities/ID1"
```



Managed identities for a VM using Azure CLI

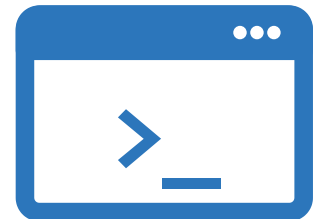
Provide an Azure AD identity for the VM

```
az identity create -g myResourceGroup -n ID1
```

Assign the identity to the VM

```
az vm identity assign -g myResourceGroup -n myVM --identities
```

```
"/subscriptions/<SUBSCRIPTION ID>/resourcegroups/<RESOURCE  
GROUP>/providers/Microsoft.ManagedIdentity/userAssignedIdentities/ID1"
```



Review

- Claims-based authorization.
- Role-based access control (RBAC) authorization.
- Virtual machine access control.

