**Azure Identity and Access Management tasks**

**Practical Task 1: Introduction to Microsoft Entra ID**

Create a basic Microsoft Entra ID setup for an organization to manage identity and access.

**Requirements:**

1. Create a new Microsoft Entra ID tenant.

2. Add at least two users to the directory.

3. Create two groups named **Developers** and **Admins**.

4. Assign the users to appropriate groups.

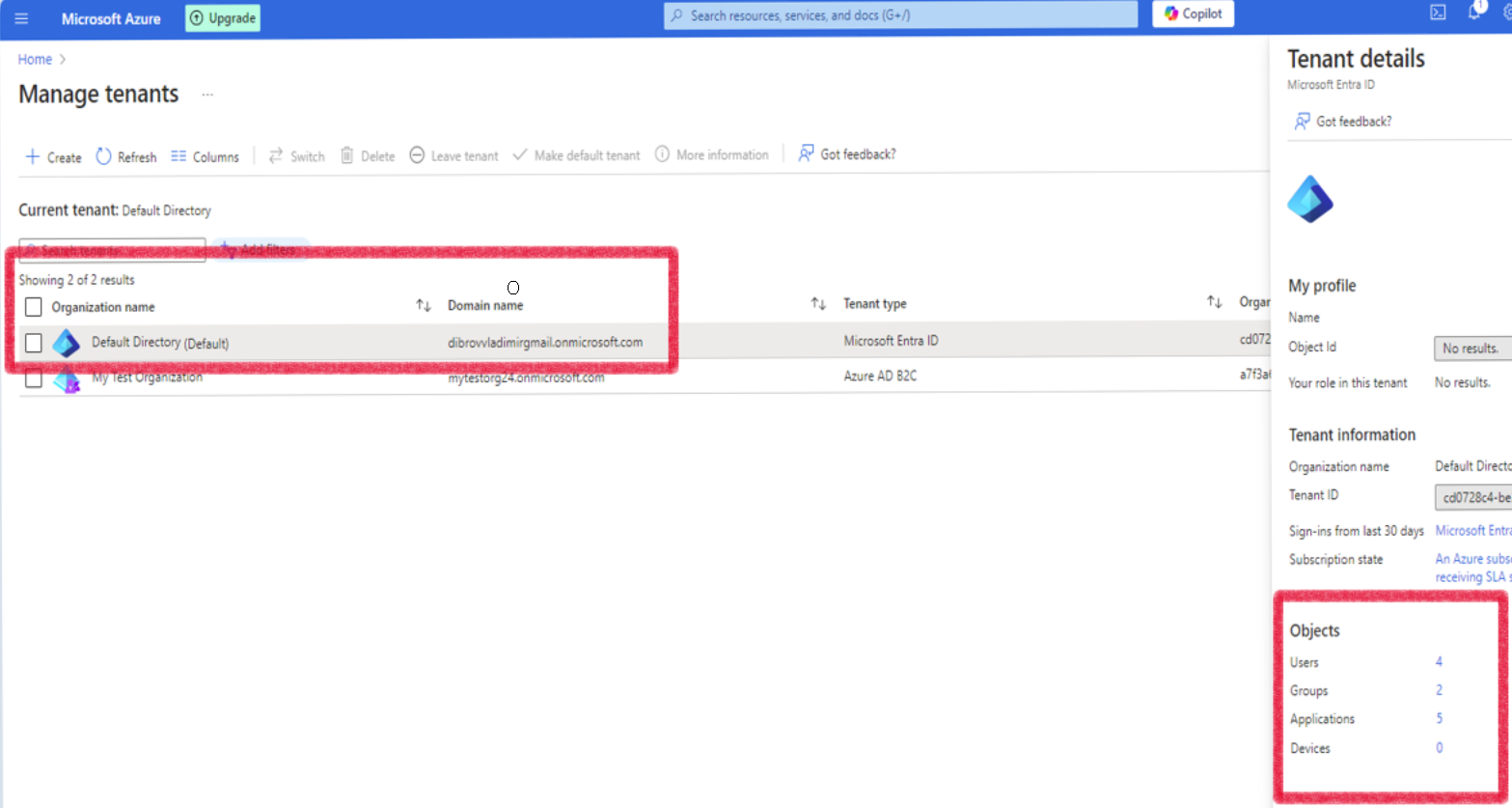
5. Assign the **Global Reader** role to the **Admins** group.

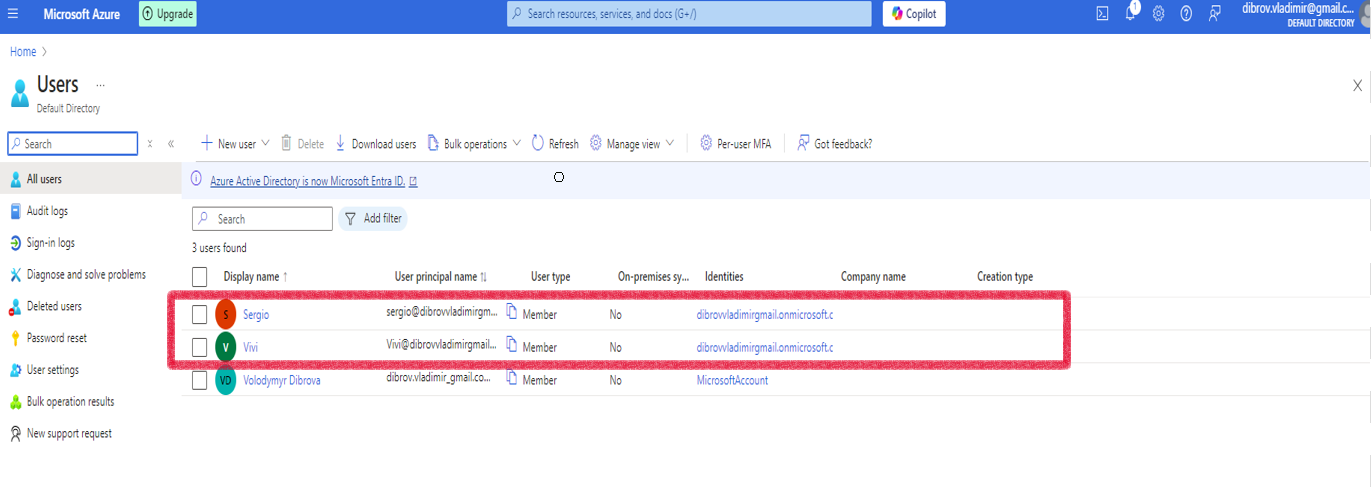
6. Assign the **Application Developer** role to the **Developers** group.

7. Verify that the role assignments function as expected for both groups.

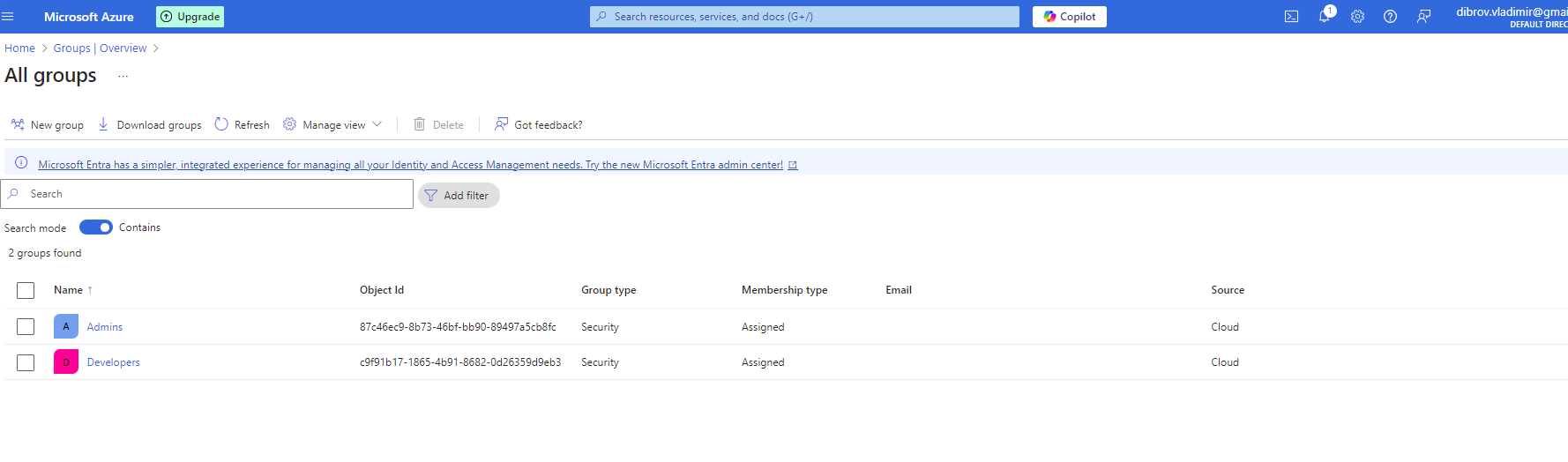
**Actions Taken:**

* Used Default Microsoft Entra ID instead of creating a new tenant due to issues with phone number registration.



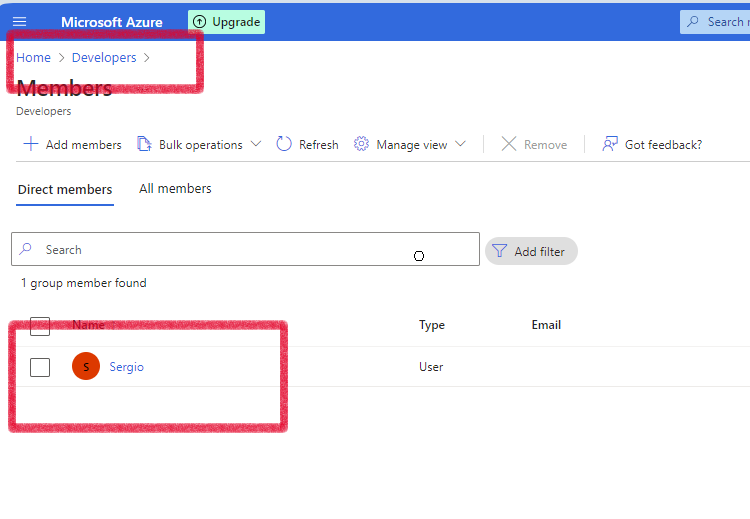
* Added two users 
* Created two groups:

1. **Developers**
2. **Admins**

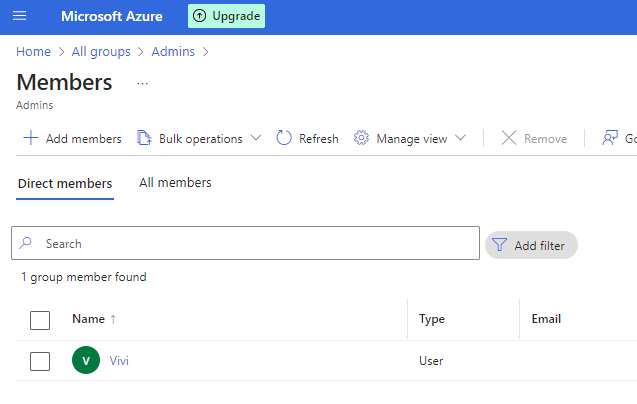


* Assigned users to the appropriate groups:

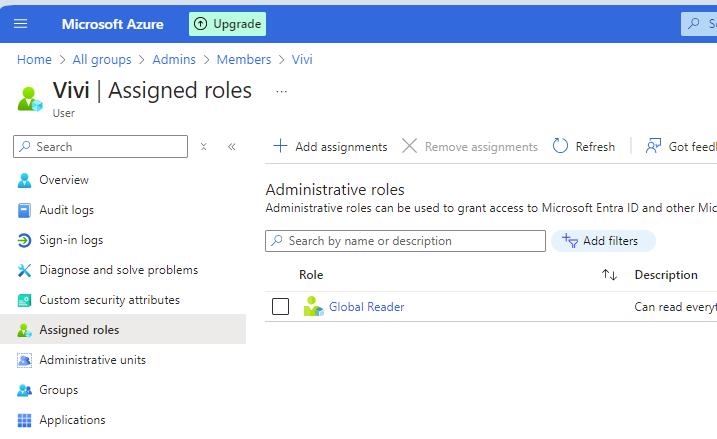
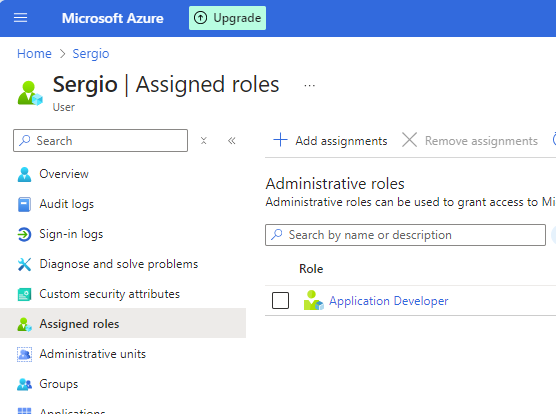
1. Developers: Sergio



1. Admins: Vivi

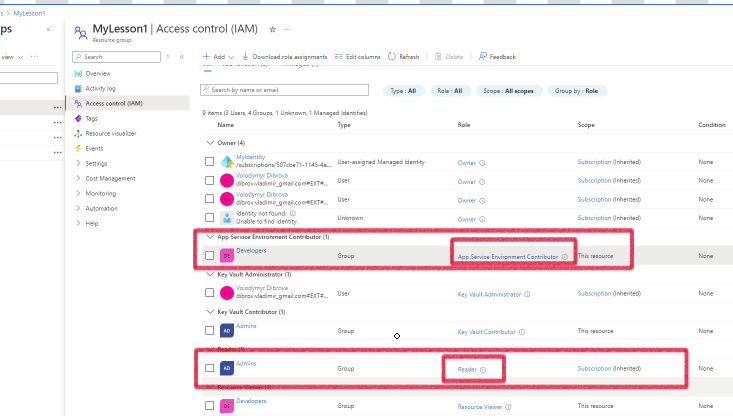


* Assigned roles:
* **The first version**



* **The second version:**

1. **Reader** role was assigned to the Admins group instead of the **Global Reader** role
2. **App Service Environment Contributor** role was assigned to the Developers group instead of the **Application Developer** role.



Verified test(Try to create a new resource by user Vivi)



**Implementation Highlights:**

* Using Default Entra ID allowed bypassing registration restrictions and successfully completing the task.

**Practical Task 2: Enabling Single Sign-On (SSO) and Multi-Factor Authentication (MFA)**

Configure Single Sign-On (SSO) and Multi-Factor Authentication (MFA) for users in a Microsoft Entra ID

directory to enhance identity and access security.

**Requirements:**

1. Enable Single Sign-On (SSO) for your Microsoft Entra ID tenant.

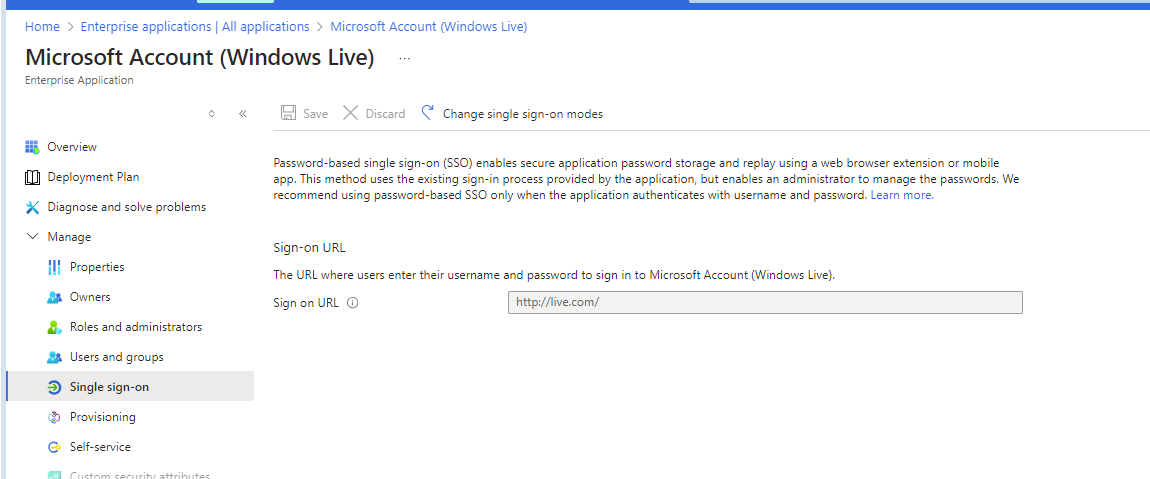
2. Enforce Multi-Factor Authentication (MFA) for all users in the directory.

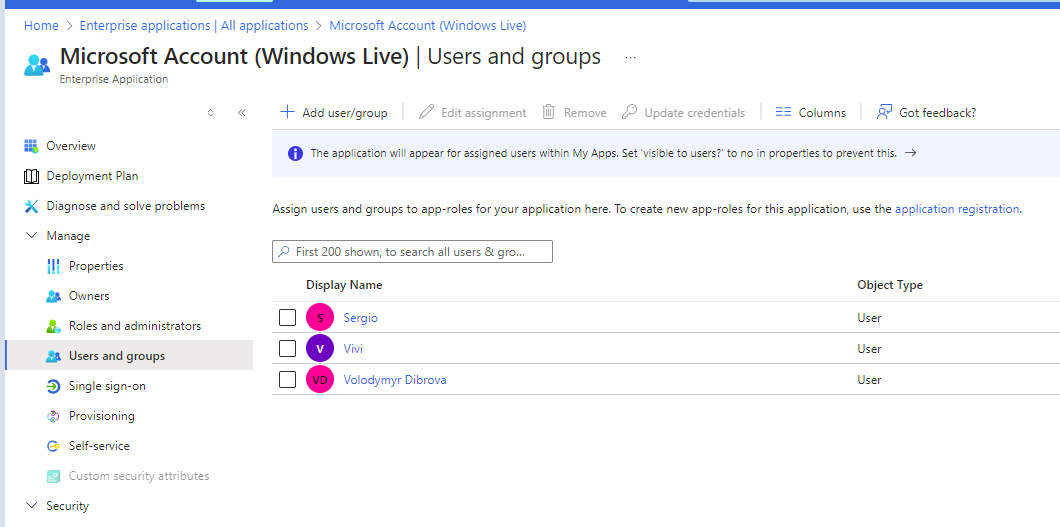
3. Configure conditional access policies to require MFA for high-risk sign-ins.

4. Verify that SSO and MFA settings are correctly applied for the users.

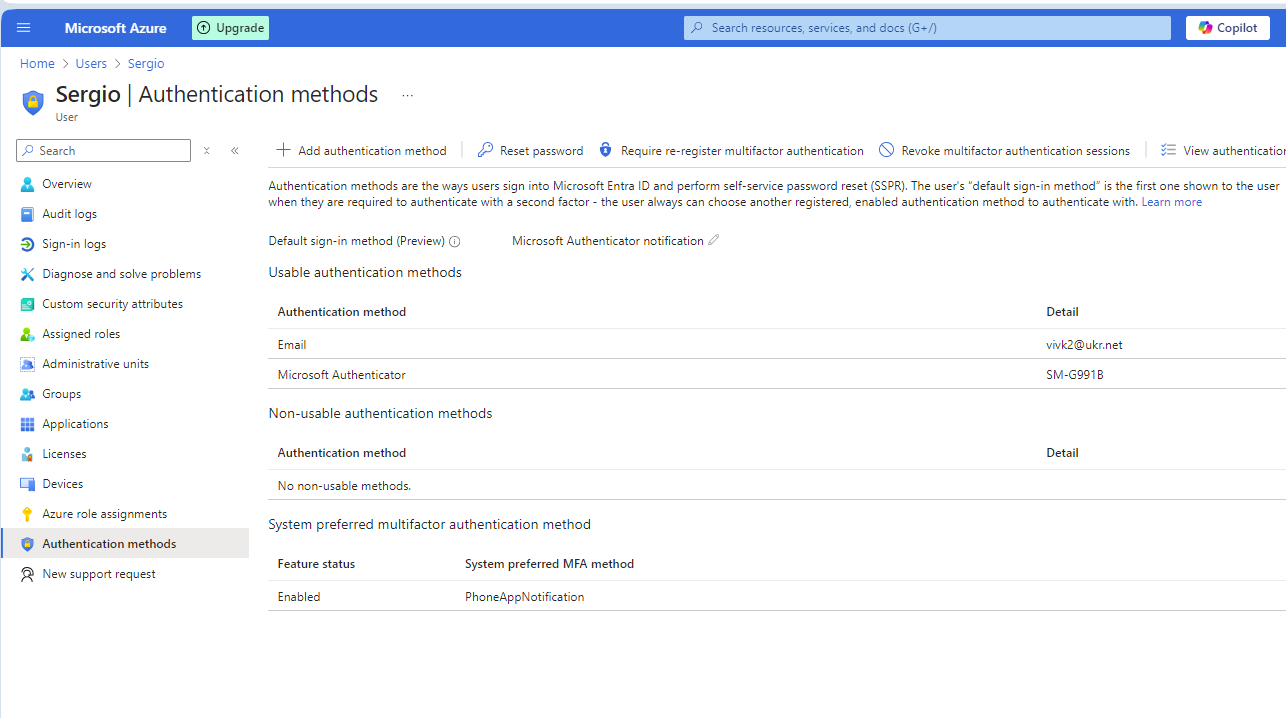
#### Actions Taken:

1. **Single Sign-On (SSO)** by **Password-based** was successfully enabled for Microsoft Entra ID.





1. **Multi-Factor Authentication (MFA)** was activated for all users in the directory.



Configure conditional access policies to require MFA for high-risk sign-ins.

**Due to registration limitations for a Premium P2 license (mobile phone number issue), the configuration was performed manually without Premium P2**

· Configured **MFA status as Enabled** for all users in the directory.

· Navigated to **Azure Active Directory > Security > Conditional Access > Named Locations** to define trusted IP addresses (currently unavailable).

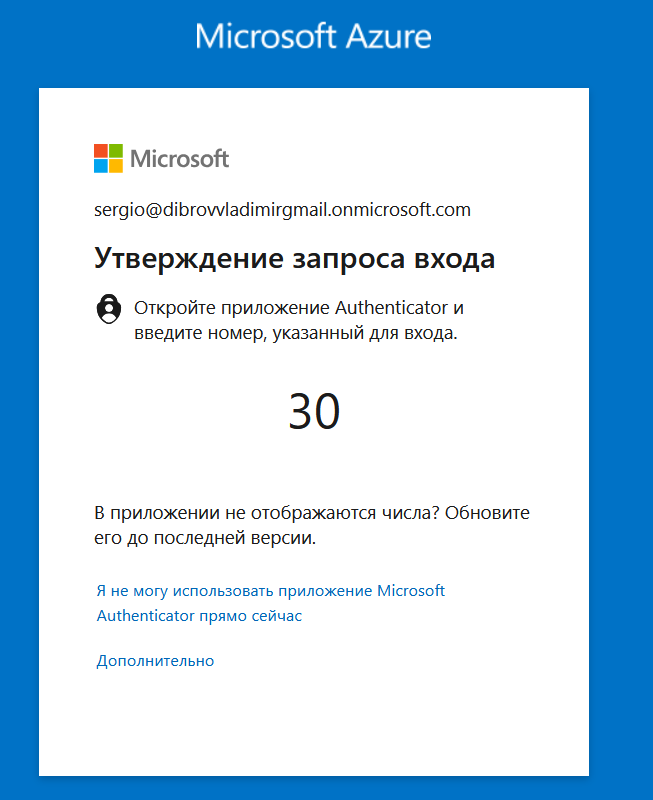
· Ensured MFA is configured for each application individually to enhance security for high-risk sign-ins.

· Used **Subscriptions > Access Control (IAM) > Add Role Assignment** to assign roles requiring MFA for the necessary resources.

#### Results:

* Achieved a manual setup approximating **high-risk sign-in** scenarios through customized configurations.
* Secured access with **SSO and MFA** functioning effectively.

1. Verification completed:
   1. SSO and MFA are functioning as expected, enhancing access security.



**Practical Task 3: Implementing Role-Based Access Control (RBAC)**

Implement Role-Based Access Control (RBAC) in Azure to manage access to resources based on roles and

ensure fine-grained access management.

**Requirements:**

1. Create a custom role named **Resource Viewer** with read-only permissions for a specific resource

group.

2. Assign the **Resource Viewer** role to the **Developers** group created earlier.

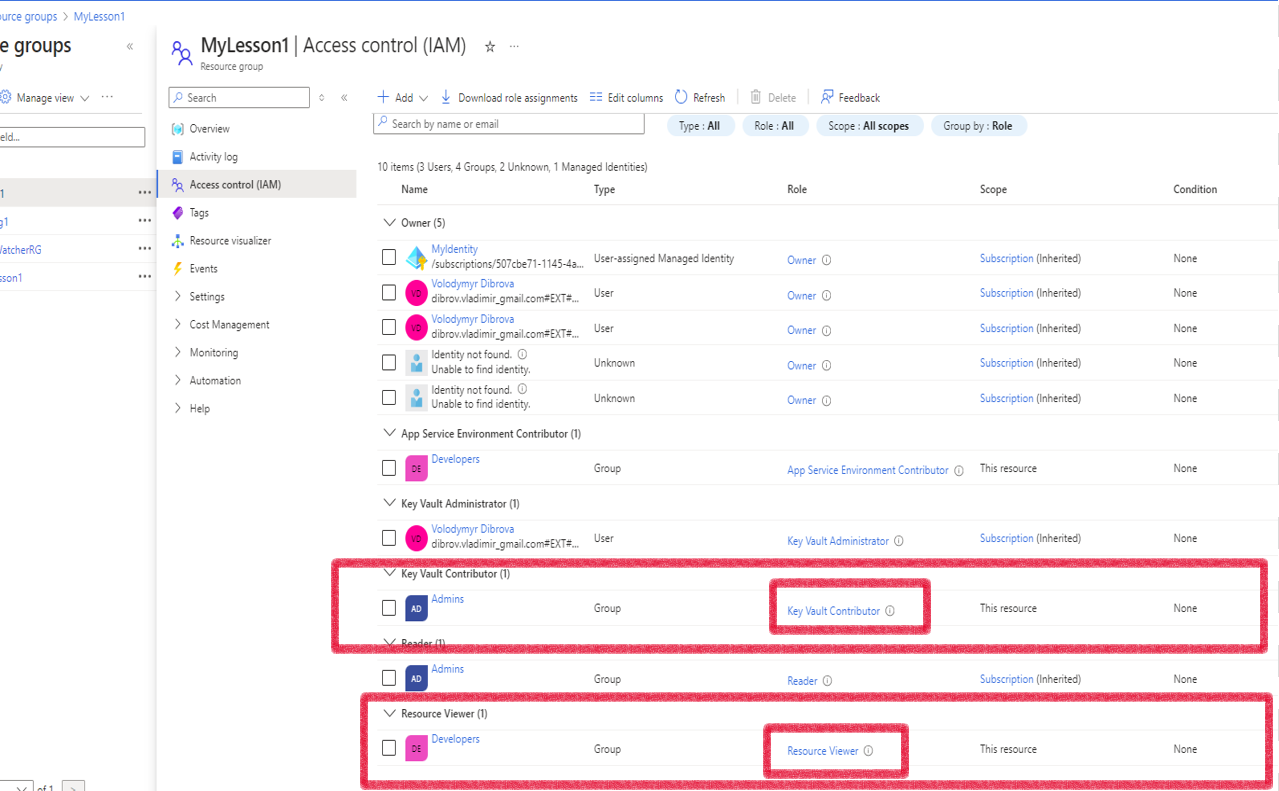
3. Assign the built-in **Contributor** role to the **Admins** group for the same resource group.

4. Verify that members of the **Developers** group have only read access and members of the **Admins**

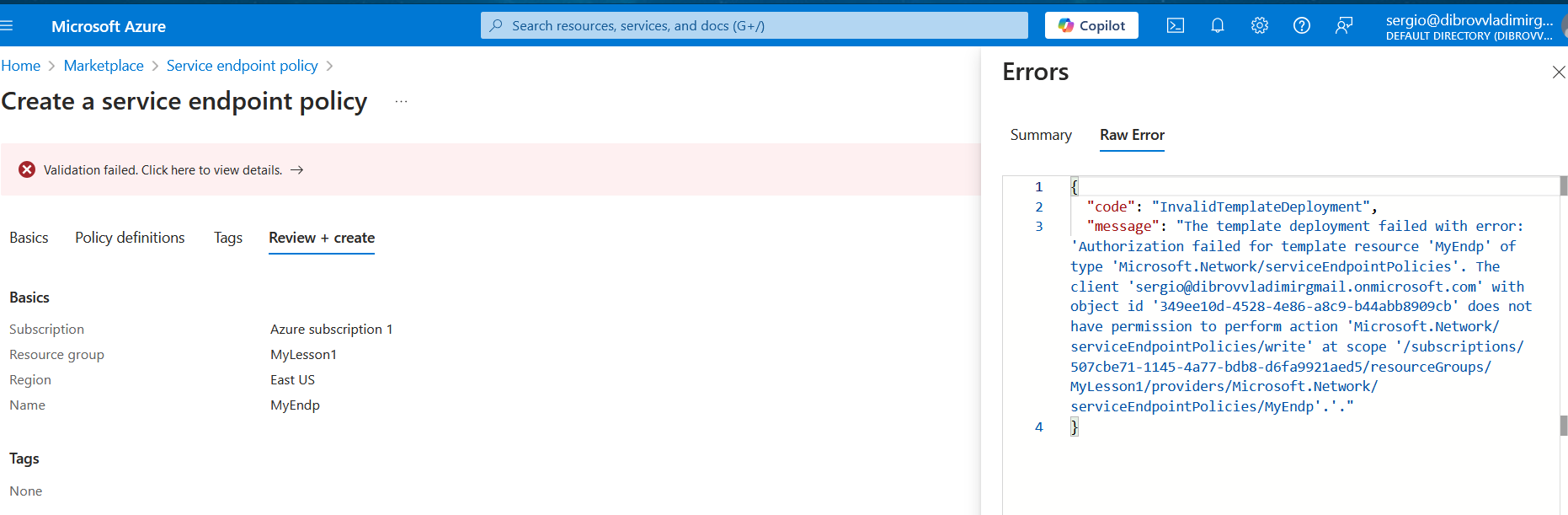
group have full access to the resource group.

#### Actions Taken:

1. Created a **custom role Resource Viewer** with read-only permissions for a specific resource group.
2. Assigned the **Resource Viewer** role to the **Developers** group created earlier.
3. Assigned the built-in **Contributor** role to the **Admins** group for the same resource group.



1. Verified access:
   1. Members of the **Developers(Sergio)** group have read-only access.
   2. Members of the **Admins** group have full access to the resource group.



#### Results:

* Successfully configured **Role-Based Access Control (RBAC)** to manage resource access based on predefined roles.

**Practical Task 4: Securing Sensitive Information with Azure Key Vault**

Set up Azure Key Vault to securely store and manage sensitive information such as keys, secrets, and

certificates.

**Requirements:**

1. Create a new Azure Key Vault in your subscription.

2. Add a secret to the Key Vault (e.g., a database connection string).

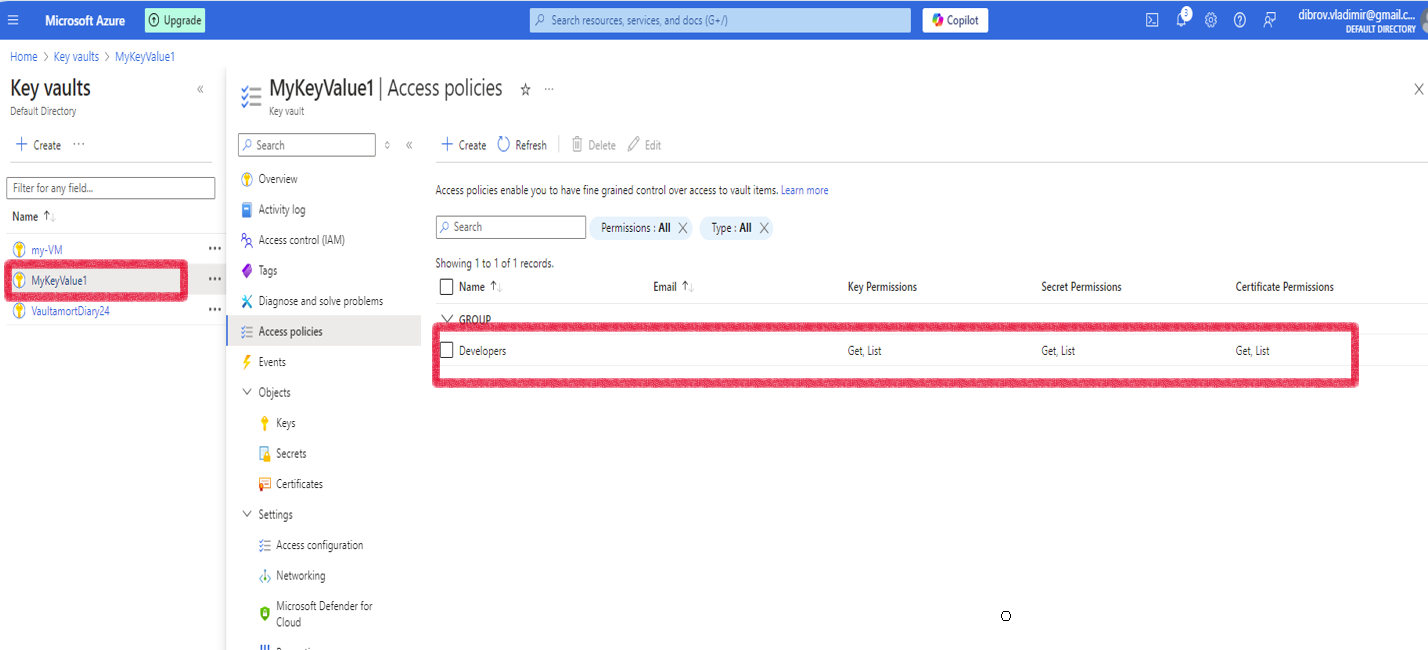
3. Set access policies to grant the **Application Developer** role (assigned to the **Developers** group)

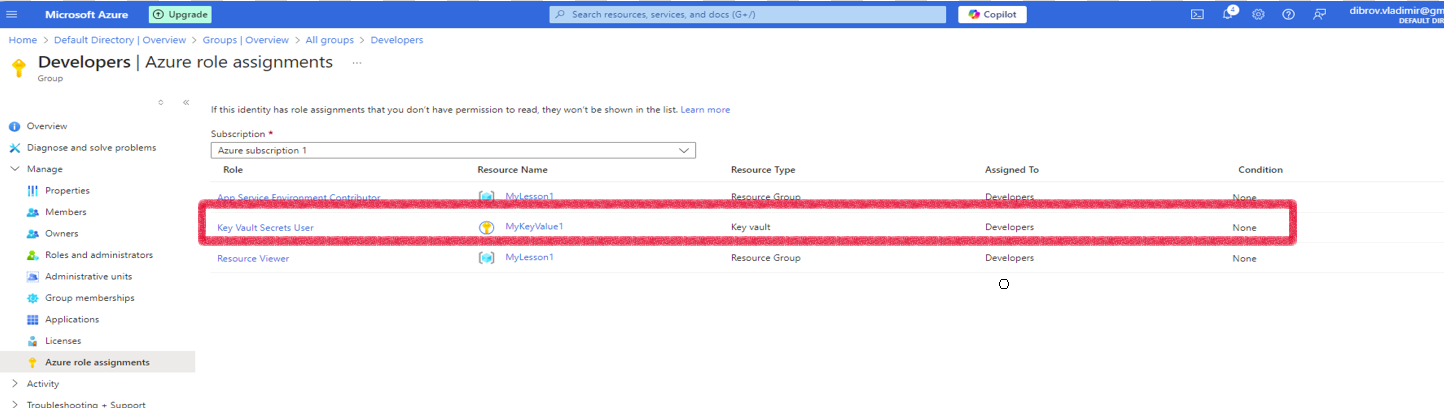
permission to retrieve secrets from the Key Vault.

4. Verify that only members of the **Developers** group can access the stored secret.

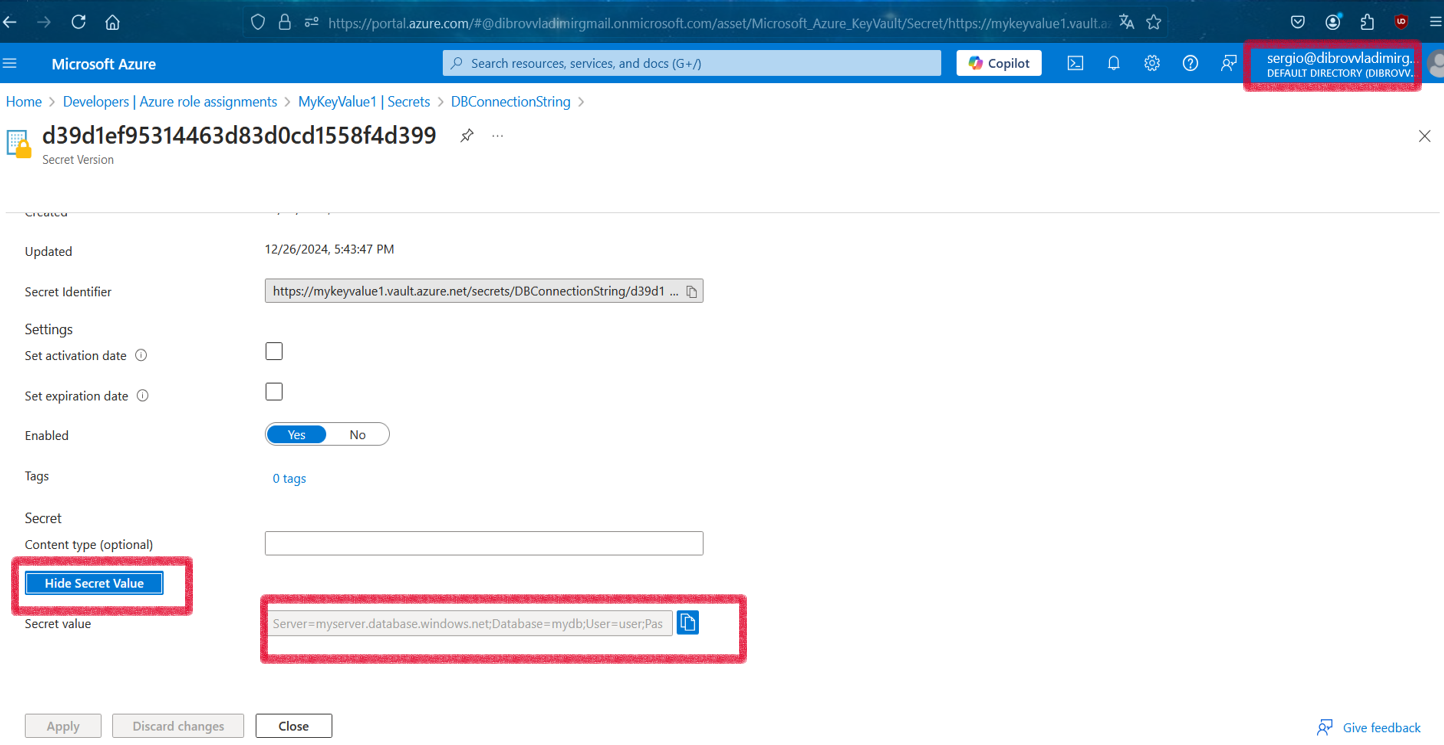
#### Actions Taken:

1. Created a new **Azure Key Vault** within the subscription.
2. Added a secret to the Key Vault (database connection string).
3. Configured access policies to grant the **Application Developer** role (assigned to the **Developers** group) permission to retrieve secrets from the Key Vault.





1. Verified access:
   1. Only members of the **Developers** group can access the stored secret.



#### Results:

* Successfully set up **Azure Key Vault** to securely store and manage sensitive information, ensuring restricted access based on roles.

**Practical Task 5: Creating and Assigning Basic Azure Policies**

Define and assign Azure Policies to enforce compliance with organizational standards for resource

management.

**Requirements:**

1. Create an Azure Policy to enforce tagging for all newly created resources with a specific tag.

2. Assign the policy to a resource group.

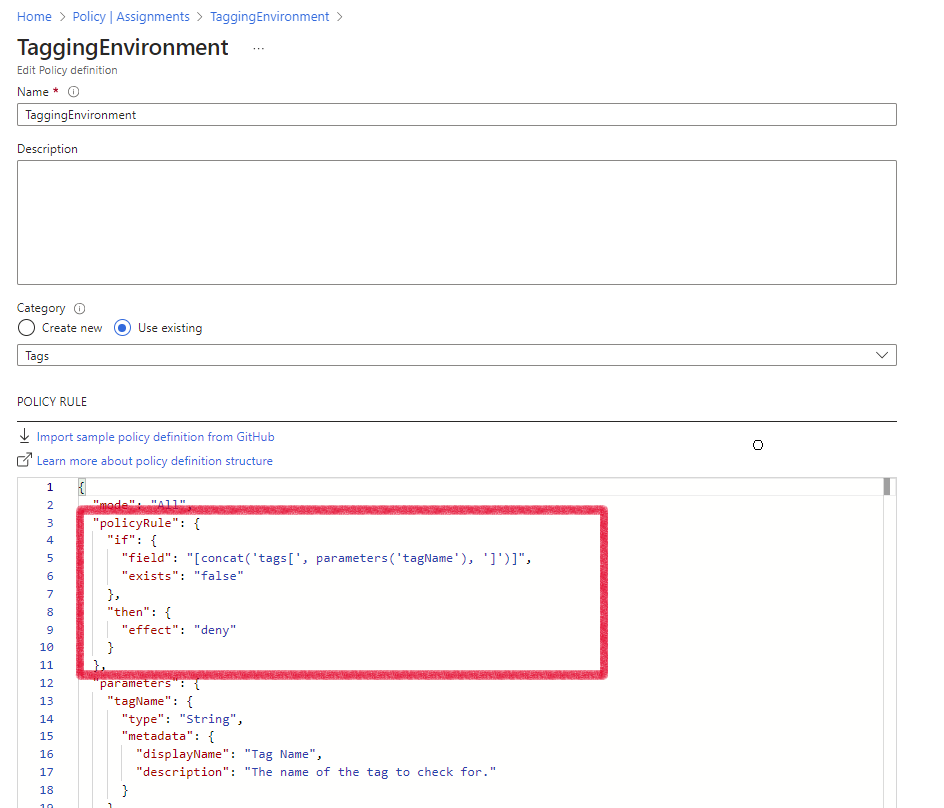
3. Verify that any new resource created in the resource group without the required tag is marked as

non-compliant.

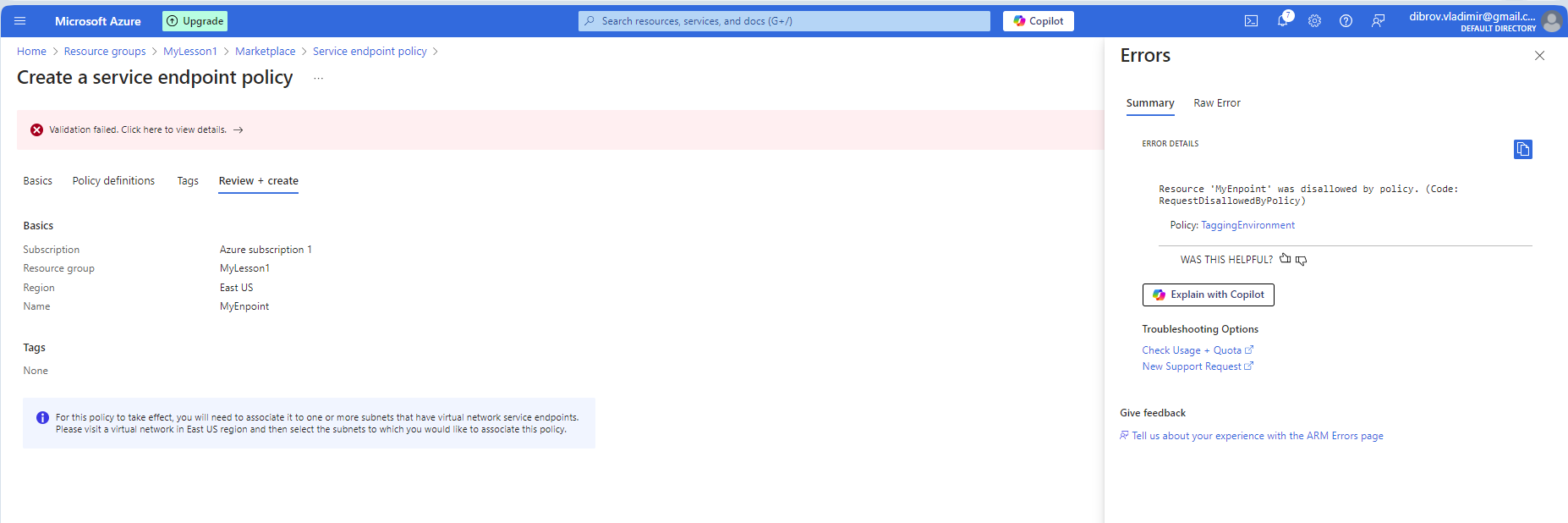
4. Review and document the compliance status of the resource group

#### Actions Taken:

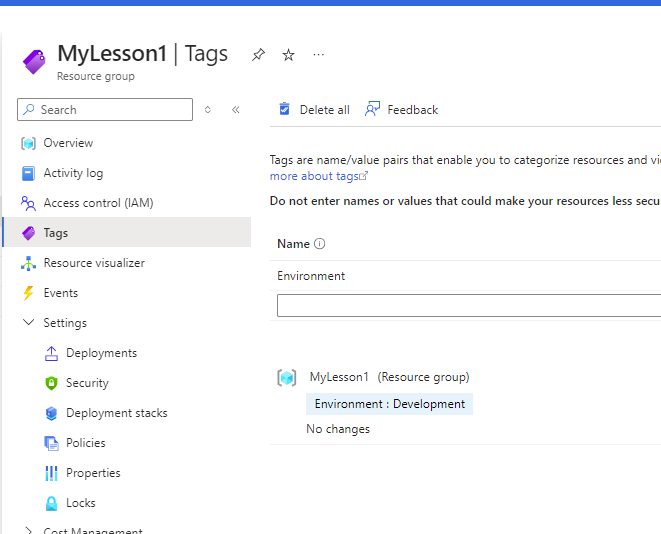
1. Created an **Azure Policy** to enforce tagging for all newly created resources with a specific tag **Environment: Development**.
2. Assigned the policy to a resource group.

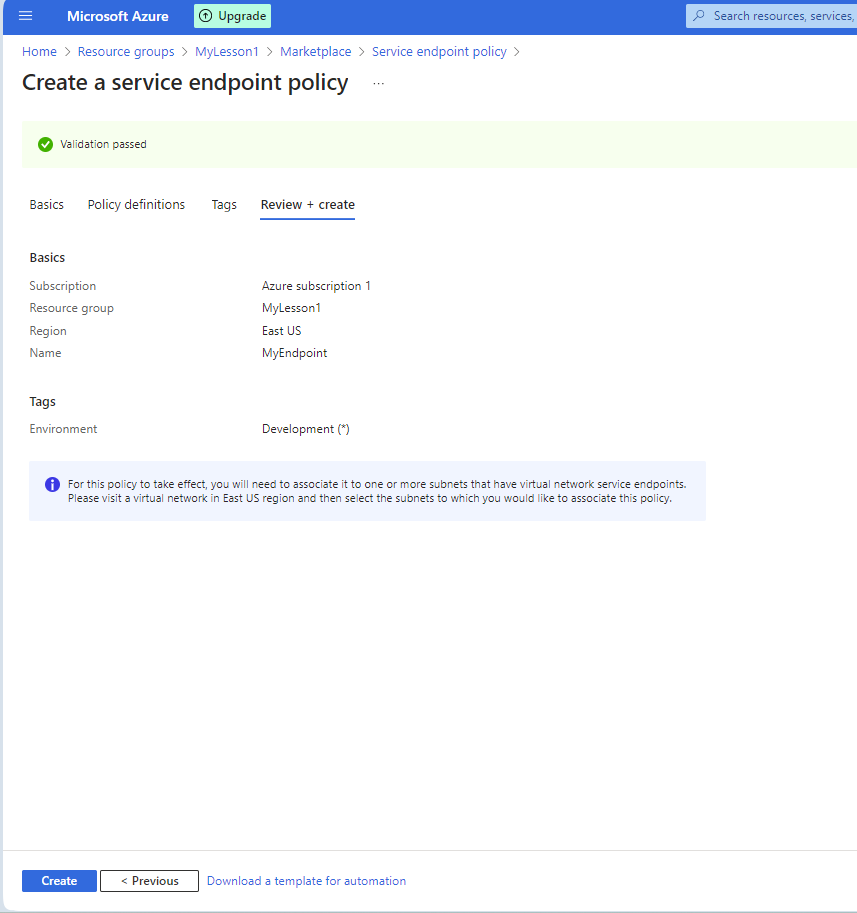


1. Verified to create resource without tag “Environment”



1. Verified to create resource with tag “Environment”





#### Results:

* Successfully implemented an **Azure Policy** to ensure compliance with organizational standards for resource management.

**Practical Task 6: Using Policy Effects to Enforce Compliance**

Configure Azure Policies with different policy effects to enforce compliance and manage resources

according to organizational standards.

**Requirements:**

1. Create a policy with the **Audit** effect to monitor and log untagged resources within a resource

group.

2. Create a policy with the **DeployIfNotExists** effect to automatically add a specific tag (Owner: IT) to

any newly created resource.

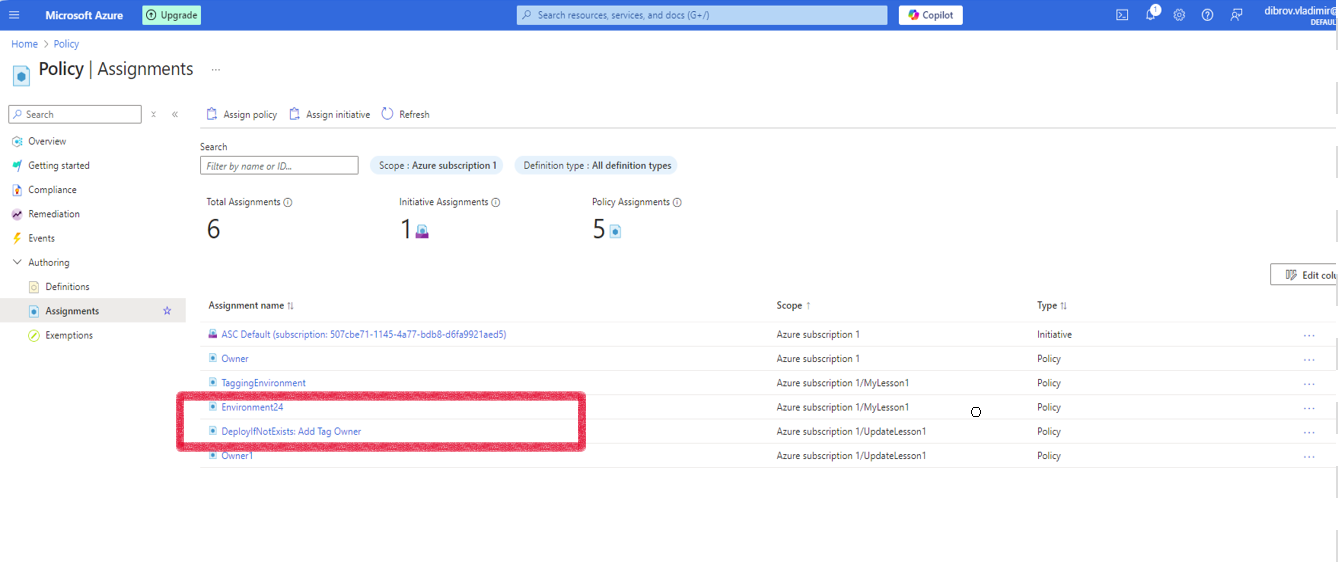
3. Assign these policies to a resource group and verify their behavior by:

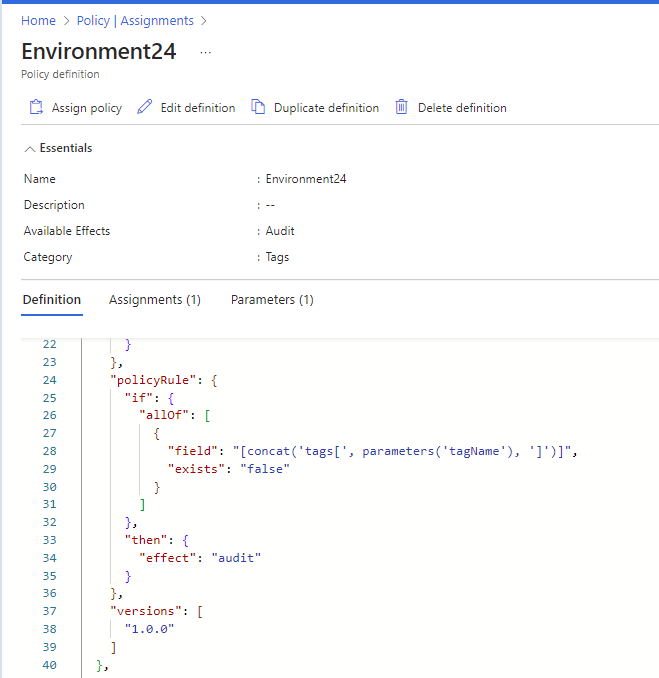
o Creating a resource without a tag and checking the compliance logs.

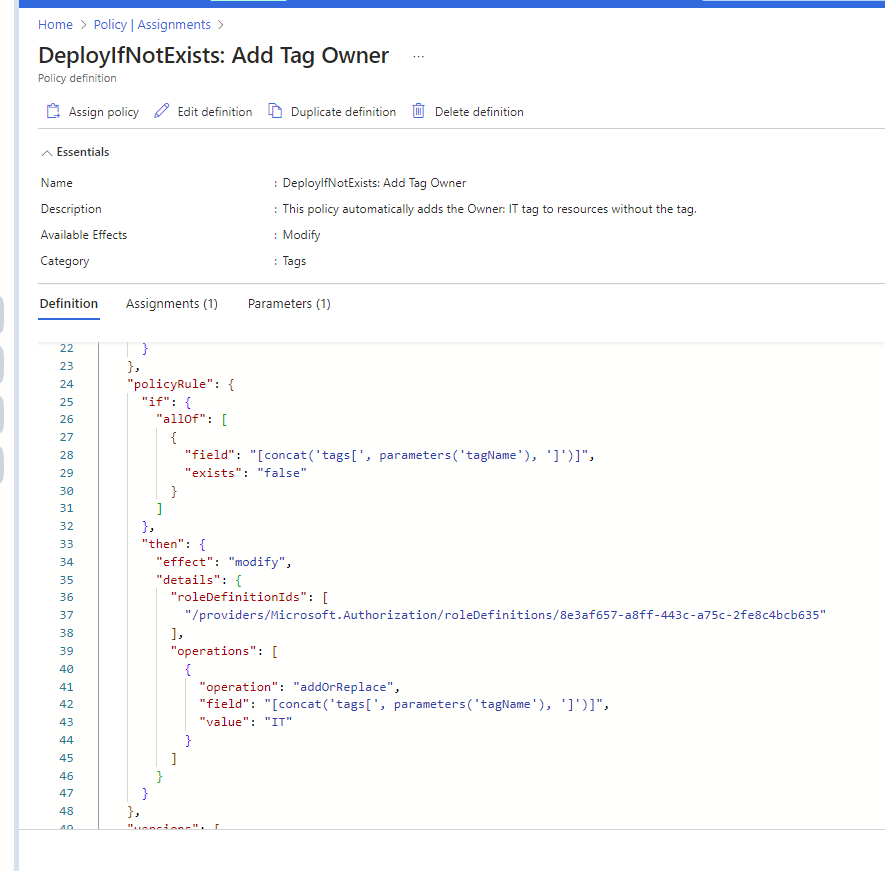
o Creating a resource to validate the automatic tag deployment.

#### Actions Taken:

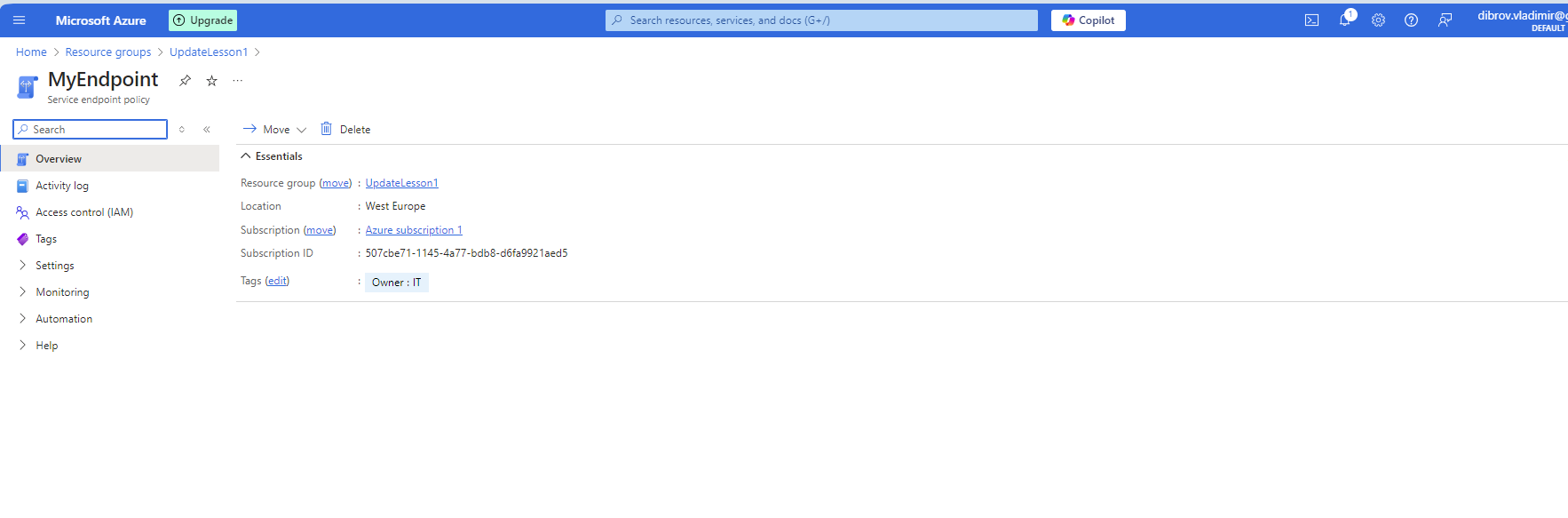
1. Created a policy with the **Audit** effect to monitor and log untagged resources within a resource group.
2. Created a policy with the **DeployIfNotExists** effect to automatically add a specific tag (**Owner: IT**) to any newly created resource.







1. Assigned both policies to a resource group and verified their behavior:
   1. Created a resource without a tag and checked the compliance logs for auditing.
   2. Created a resource to confirm the automatic deployment of the specified tag.



#### 

#### Results:

* Successfully configured and tested **Azure Policies** with different effects.