



Study Definition Repository (SDR) Reference Implementation

Setup and Deployment Guide Version 6.0

Disclaimer

These materials and information, as well as the underlying code/application they relate to are provided by TransCelerate Biopharma Inc. AS IS. Any party using or relying on this information and, these materials and/or the underlying code/application do so entirely at their own risk. Neither TransCelerate nor its members will bear any responsibility or liability for any harm, including indirect or consequential harm, that a user may incur from use or misuse of this information, materials, or underlying code/application.

TransCelerate does not endorse any particular software, system, or service. And the use of specific brands of products or services by TransCelerate and its collaboration partners in developing the SDR Reference Implementation should not be viewed as any endorsement of such products or services. To the extent that the SDR Reference Implementation incorporates or relies on any specific branded products or services, this resulted out of the practical necessities associated with making a reference implementation available to demonstrate the SDR's capabilities.

Document History

Version No.	Date	Author	Revision Description
V1.0	15-Mar-2022	ACN	Initial Draft
V2.0	29-Jun-2022	ACN	Removed Smoke Testing section
V3.0	19-Aug-2022	ACN	Updated Section 2.8 manual configuration changes and Section 5 APIM setup as per the latest infra changes.
V4.0	03-Apr-2023	ACN	Updated Section 3.1 Low Level Design Document, 2.8 manual configuration changes and Section 5 APIM setup as per the latest infra changes. Headers of section 2.8.1 and section 5 updated.
V5.0	09-Jun-2023	ACN	Updated Low Level Design Document in section 3.1. Added Function App Delegated subnet verification screenshots in section 3.4. Updated Header of section 4.3. Updated section 4.4 with latest UI and Function App deployment verification screenshots.
V6.0	17-Oct-2023	ACN	Added APIM Developer Portal configuration.

Table of Contents

1. Introduction	7
1.1. Definitions and Acronyms.....	7
1.2. Document Scope.....	8
1.3. Out of Scope	8
1.4. Audience	8
1.5. Pre-Requisites.....	8
2. Azure Infrastructure	8
2.1. Resource Provider Registration	8
2.2. Create Azure AD Groups	11
2.3. Create Users.....	14
2.4. Role Based Access Controls (RBAC).....	14
2.5. Storage Account and Service Principal Configuration in Azure	17
2.6. Adding Secrets in GitHub.....	19
2.7. Execute GitHub Action for IaC deployment	21
2.8. Manual Configuration Changes on Azure Platform.....	22
2.8.1. OAuth 2.0 Configuration for SDR UI Application.....	22
2.8.2. Key Vault	26
2.8.3. Storage Account	31
3. Resource Validation.....	31
3.1. Low-Level Design Document	31
3.2. Virtual Network	33
3.3. Subnet	36
3.4. Delegated Subnet.....	37
3.5. Other Resources	40
4. Application Code Deployment.....	40
4.1 GitHub Secrets used in Workflow	41
4.2 Deploy the UI Application	42
4.3 Deploy Back-End API and Function App.....	43
4.4 Deployment Verification	46
5. APIM Setup	50
5.1. Developer Portal Configuration	60
5.1.1. Creating a Product.....	61
5.1.2. Azure AD Group Creation.....	62
5.1.3. Add Azure AD Group in APIM	64

5.1.4. Add Azure AD Group in Product.....	64
5.1.5. Add App registration for Azure AD login	66
5.1.6. Identities in APIM	67
5.1.7. Change Publisher Name in APIM.....	67
5.1.8. Add User to the AD Group for Developer Portal Access	68

List of Figures

Figure 1 Select Subscriptions in Azure Portal	9
Figure 2 Select Subscription	9
Figure 3 Resource Providers in a Subscription.....	10
Figure 4 Registering Resource Providers.....	11
Figure 5 Adding new groups	13
Figure 6 Add New Group	13
Figure 7 Create New User	14
Figure 8 Access Control (IAM)	15
Figure 9 Add Role Assignment.....	15
Figure 10 Select Roles.....	16
Figure 11 Select Members for Role Assignment	16
Figure 12 View Role Assignments.....	17
Figure 13 GitHub Actions Secrets	20
Figure 14 Add new GitHub Action Secret.....	21
Figure 15 Azure AD - App Registration	22
Figure 16 Azure AD - App Registration - Redirect URI.....	23
Figure 17 Azure AD - App Registration - Expose an API.....	23
Figure 18 Azure AD - App Registration - API Permission	24
Figure 19 Azure AD - App Registration - Add API Permission.....	24
Figure 20 Azure AD - App Registration - API Permission - Grant Admin Consent.....	25
Figure 21 Azure AD - App Registration - Certificates & Secrets	25
Figure 22 Azure AD - App Registration - Essential Secrets.....	26
Figure 23 Add Key Vault Access Policy.....	28
Figure 24 Select Secret Permissions in Access Policy in Key Vault	29
Figure 25 Select Principal (List of users) In Key Vault Access Policy	29
Figure 26 Create Secrets in Key Vault	30
Figure 27 Add Secrets values in Key Vault	30
Figure 28 Disable public access for Blob storage.....	31
Figure 29 Resource Naming Convention.....	32
Figure 30 SDR Resource Naming Convention	32
Figure 31 VNet Configurations	33
Figure 32 Virtual Network.....	34
Figure 33 Virtual Network - DDoS protection.....	34
Figure 34 Virtual Network - Tags.....	35
Figure 35 Virtual Network - Diagnostic Setting	35
Figure 36 Subnet Details.....	36
Figure 37 Subnet Details.....	37
Figure 38 Delegated Subnet-001 Details.....	38
Figure 39 Delegated Subnet-001 Service Endpoints Details	38
Figure 40 Delegated Subnet-002 Details.....	38
Figure 41 Delegated Subnet-002 Service Endpoints Details	39
Figure 42 Function App Delegated Subnet-002 Details	39
Figure 43 Function App Delegated Subnet-002 Service Endpoints Details	40

Figure 44 SDR UI Repo - GitHub Actions.....	42
Figure 45 GitHub Actions - CI Workflow	42
Figure 46 GitHub - CI Workflow Run	43
Figure 47 GitHub CI Workflow Output	43
Figure 48 GitHub SDR API Repo	44
Figure 49 GitHub Actions CI Worklfow	44
Figure 50 GitHub CI Workflow Run	45
Figure 51 GitHub CI Workflow Run	45
Figure 52 GitHub CI Workflow Output	46
Figure 53 Azure Portal	46
Figure 54 Azure Portal - SDR UI App Service	47
Figure 55 SDR UI App Service - Deployment Center Details	47
Figure 56 SDR UI App Service Deployment Center -Logs	48
Figure 57 Azure Portal	48
Figure 58 Azure Portal - SDR Function App Service	49
Figure 59 SDR Function App Service – Deployment Center	49
Figure 60 SDR Function App Service Deployment Center - Logs	50
Figure 61 Client Certificate.....	51
Figure 62 APIM Certificates	51
Figure 63 APIM Certificates - Client Certificates.....	52
Figure 64 APIM Certificates - Client Certificate	52
Figure 65 APIM Inbound Processing.....	54
Figure 66 APIM - Inbound Policy.....	58

1. Introduction

This document details the steps required to set up a new environment for the Study Definition Repository (SDR) – Reference Implementation (Release V2.0) on Microsoft Azure Cloud Platform. The SDR Reference Implementation is an attempt to demonstrate the vision of the DDF initiative, not a commercial product.

To be clear, TransCelerate does not endorse any particular software, system, or service. And the use of specific brands of products or services by TransCelerate and its collaboration partners in developing the SDR Reference Implementation should not be viewed as any endorsement of such products or services. To the extent that the SDR Reference Implementation incorporates or relies on any specific branded products or services, such as Azure, this resulted out of the practical necessities associated with making a reference implementation available to demonstrate the SDR's capabilities. Users are free to download the source code for the SDR from GitHub and design their own implementations. Those implementations can be on an environment of the user's choice, and do not have to be on Azure.

This document is organized sequentially including Infrastructure, Authentication (OAuth and APIM), UI and API components setup and is presented in a dependency-based order. It provides details for Infrastructure setup, environment validation, deploying the Web Application for User Interface and Application Programming Interface.

All the sections listed here are mandatory for the environment setup.

1.1. Definitions and Acronyms

Term / Abbreviation	Definition
AAD	Azure Active Directory
AD	Active Directory
API	Application Programming Interface
APIM	Application Programming Interface Management
Azure CLI	Azure Command Line Interface
DB	Database
DDF	Digital Data Flow
HTTP	Hypertext Transfer Protocol
IaC	Infrastructure-as-Code
JSON	JavaScript Object Notation
LLD	Low Level Design
MMC	Microsoft Management Console
RBAC	Role Based Access Control
REST	Representational State Transfer
SDR	Study Definition Repository
UI	User Interface
URL	Uniform Resource Locator
VNet	Virtual Network

1.2. Document Scope

Scope of the document includes steps on how to create the Active Directory Groups, Service Connections, Service Principals, and how to configure access using RBAC for SDR RI on Azure Platform. This document also describes the process of SDR RI application deployment using GitHub actions for both UI and API, along with platform setup for hosting and integrating UI application (WebApp), Web API, APIM and CosmosDB on Azure Platform. This also captures the Environment validation steps and Application Smoke testing.

1.3. Out of Scope

This document does not mention any details regarding setting up of a new Cloud Subscription as this assumes there is already an active subscription. This document also does not address application architecture patterns or designs.

1.4. Audience

This document assumes a good understanding of Azure concepts and services. The audience for this document includes Azure Administrators, DevOps Engineers, Developers with experience in Angular and .NET development.

1.5. Pre-Requisites

- Access to the DDF SDR [low-level design document](#).
- Access to the [azure portal](#) with required permissions (detailed in upcoming sections).
- Azure CLI. Refer to [Install CLI](#)
- User Should have Repo Admin level of access to add/replace the GitHub secrets in GitHub.
- An Active Azure Tenant and Subscription. To setup Azure subscription please follow the below Microsoft Documentation
[Create your initial Azure subscriptions - Cloud Adoption Framework | Microsoft Docs](#)

2. Azure Infrastructure

2.1. Resource Provider Registration

GOAL:

The Resource Provider Registration configures the Azure subscription to work with the resource provider.

PRE-REQUISITES:

Global Admin or Subscription Owner level of access to Azure.

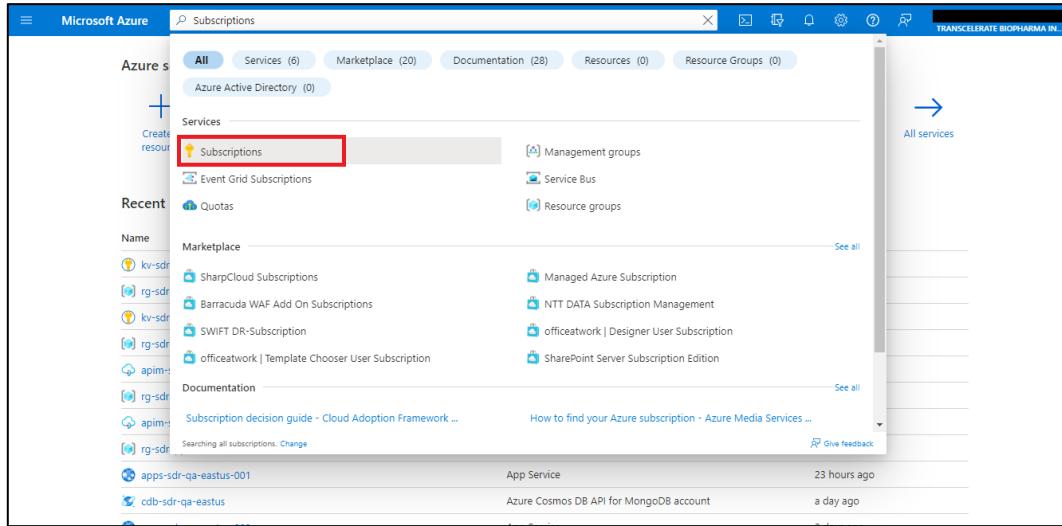
STEPS:

- i. Sign into the Azure portal.

ii. On the Azure portal menu

- Search for Subscriptions.
- Select it from the available options as shown below.

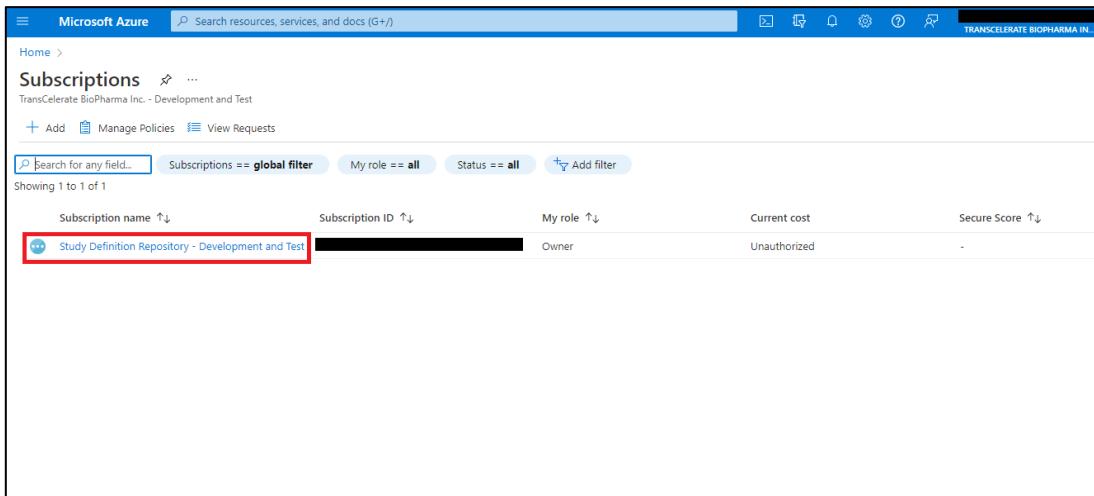
Figure 1 Select Subscriptions in Azure Portal



Name	Description	Last Activity
kv-sdr	SharpCloud Subscriptions	23 hours ago
rg-sdr	Barracuda WAF Add On Subscriptions	23 hours ago
kv-sdr	SWIFT DR-Subscription	23 hours ago
rg-sdr	officeatwork Template Chooser User Subscription	23 hours ago
apim-rg-sdr	Subscription decision guide - Cloud Adoption Framework ...	How to find your Azure subscription - Azure Media Services ...
rg-sdr	Searching all subscriptions. Change	Give feedback
apps-sdr-qa-eastus-001	App Service	23 hours ago
cdb-sdr-qa-eastus	Azure Cosmos DB API for MongoDB account	a day ago

iii. Select the subscription you want to view.

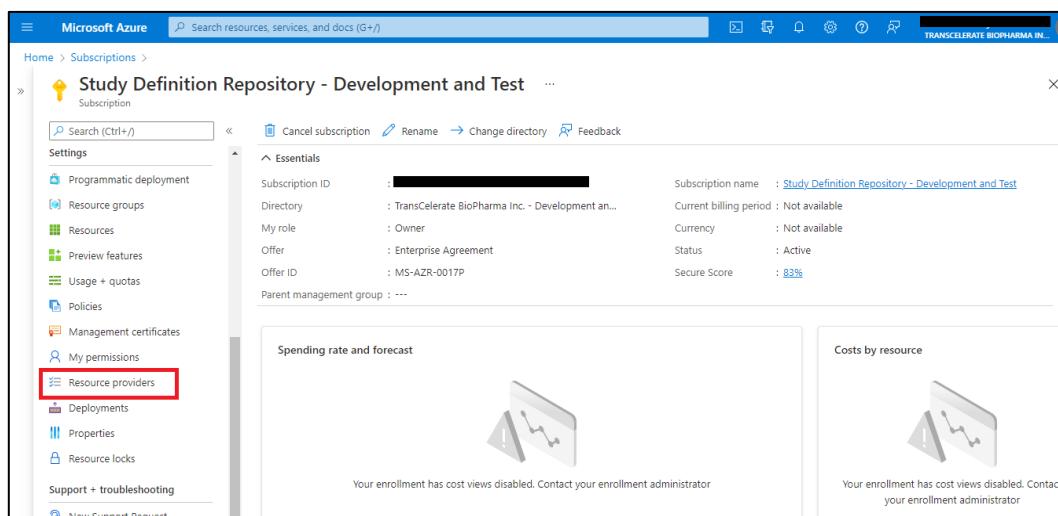
Figure 2 Select Subscription



Subscription name ↑	Subscription ID ↑	My role ↑↓	Current cost	Secure Score ↑↓
Study Definition Repository - Development and Test	[REDACTED]	Owner	Unauthorized	-

- iv. On the left menu
Under Settings
select Resource providers.

Figure 3 Resource Providers in a Subscription

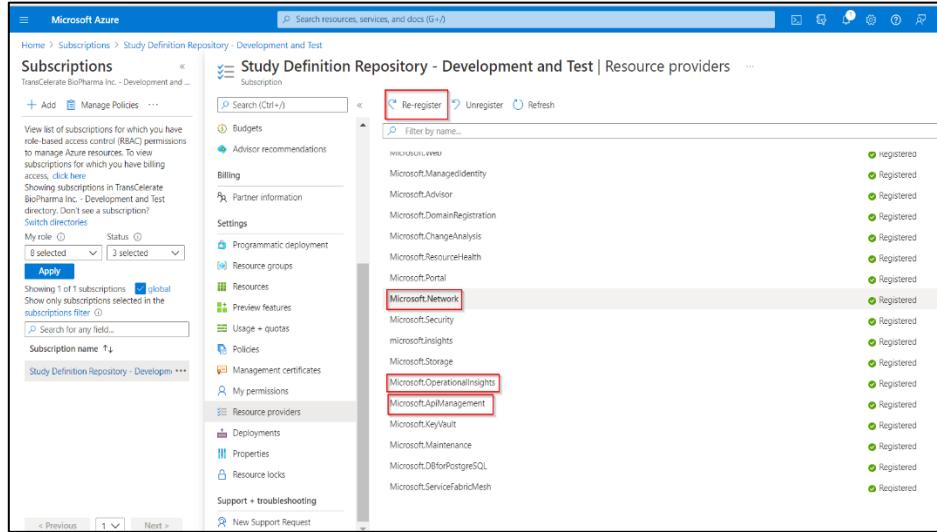


- v. Find the resource provider you want to register and select Register. To maintain least privileges in your subscription, only register the additional resource providers (other than default) that are required as listed below.

The Providers to be registered are:

- Microsoft.Network
- Microsoft.OperationalInsights
- Microsoft.ApiManagement
- Microsoft.DocumentDB

Figure 4 Registering Resource Providers



2.2. Create Azure AD Groups

GOAL:

Create Azure AD groups to manage Role Based Access Controls (RBAC) on resources for team members.

PRE-REQUISITES:

- Global administrator/Owner/User Administrator level of access at Active Directory Level.

Below are the groups and role assignments created and managed for SDR Reference Implementation.

Table 1 Group and Role Assignments

RBAC Group Name	Azure Built-In Role	Scope	Usage
GlobalAdmin_Group	Global Administrator	Azure Active Directory	Can manage all aspects of Azure AD and Microsoft services that use Azure AD identities.
Contributor_Group	Contributor	Subscription	Grants full access to manage all resources but does not allow you to assign roles in Azure RBAC, manage assignments in Azure Blueprints, share image galleries, or perform Azure Policy operations.

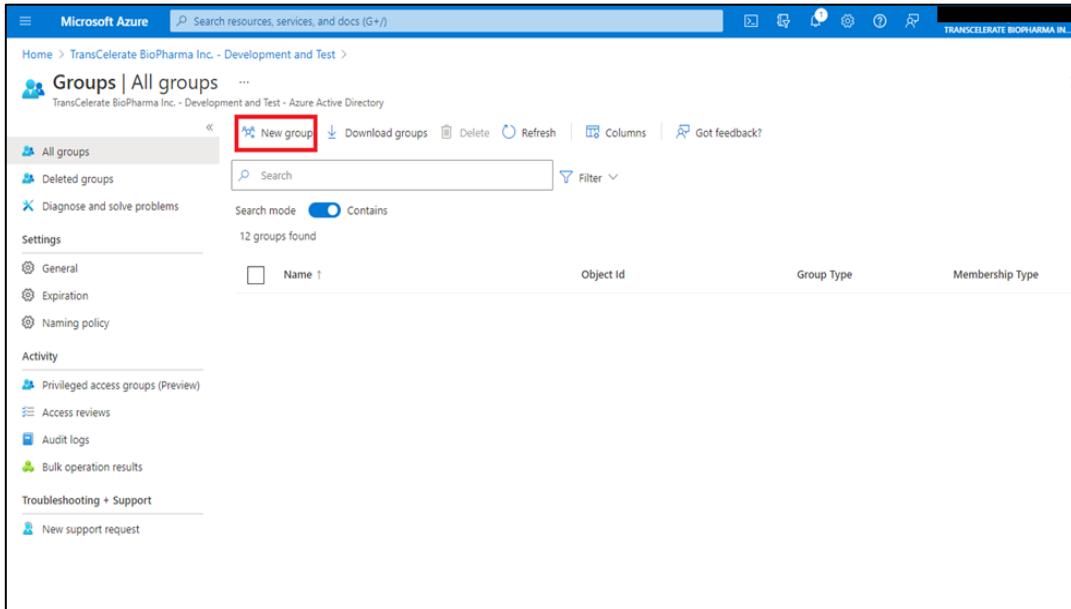
RBAC Group Name	Azure Built-In Role	Scope	Usage
Owner_Subscription_Group	Owner	Subscription	Grants full access to manage all resources, including the ability to assign roles in Azure RBAC.
Infra_Group	Contributor	Subscription	Grants full access to manage all resources but does not allow you to assign roles in Azure RBAC, manage assignments in Azure Blueprints, share image galleries, or perform Azure Policy operations.
	Global Reader	Azure Active Directory	Can be able to read all users and groups information in Azure AD.
	User Administrator	Azure Active Directory	Can manage all aspects of users and groups, including resetting passwords for limited admins.
	User Access Administrator	Subscription	Let's you manage user access to Azure resources.
DevelopmentTeam_Group	Reader	Subscription	Grants Reader access for all the resources in the Subscription but does not allow you to manage them.
	Contributor	Resource Group	Grants full access to manage all resources in the Resource Group.
TestingTeam_Group	Reader	Resource Group	Grants Reader access for all the resources in the Subscription but does not allow you to manage them.
AppRegistration_Group	Application administrator	Azure Active Directory	Can create and manage all aspects of app registrations and enterprise apps.

Note: To assign Azure AD roles to groups required Azure AD Premium P1 or P2 license, currently Azure AD Free plan has been leveraged. Follow this [documentation](#) to assign Azure AD role to groups.

STEPS:

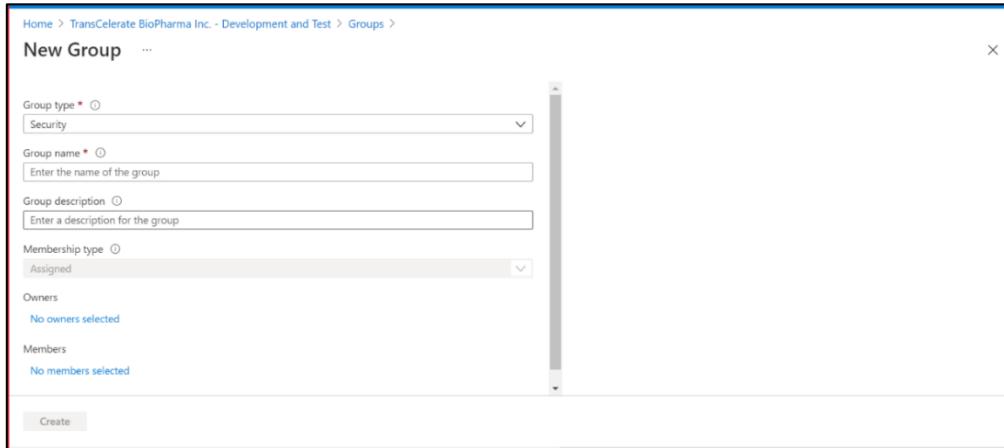
- i. Login to Azure portal
- ii. Search for Azure Active Directory (AAD)
- iii. Click on the Groups on the left panel in AAD.
- iv. Click on New group tab on the top as shown below, add the security group and save the changes by adding the members to the group.

Figure 5 Adding new groups



The screenshot shows the Microsoft Azure Groups page. The URL is "Home > TransCelerate BioPharma Inc. - Development and Test > Groups | All groups". On the left sidebar, there are sections for "All groups", "Deleted groups", "Diagnose and solve problems", "Settings" (with options for General, Expiration, and Naming policy), "Activity" (with options for Privileged access groups (Preview), Access reviews, Audit logs, and Bulk operation results), and "Troubleshooting + Support" (with a New support request link). The main area shows a table with 12 groups found. The columns are Name (sorted ascending), Object Id, Group Type, and Membership Type. At the top of the main area, there is a toolbar with "New group" (highlighted with a red box), "Download groups", "Delete", "Refresh", "Columns", and "Got feedback?". Below the toolbar is a search bar and a "Filter" dropdown.

Figure 6 Add New Group



The screenshot shows the "New Group" dialog box. The URL is "Home > TransCelerate BioPharma Inc. - Development and Test > Groups > New Group". The form fields are as follows:

- Group type *: Security (selected from a dropdown menu)
- Group name *: Enter the name of the group (text input field)
- Group description: Enter a description for the group (text input field)
- Membership type: Assigned (selected from a dropdown menu)
- Owners: No owners selected
- Members: No members selected

At the bottom right of the dialog box is a "Create" button.

2.3. Create Users

GOAL:

Create users for provisioning access to the resources on Azure Portal.

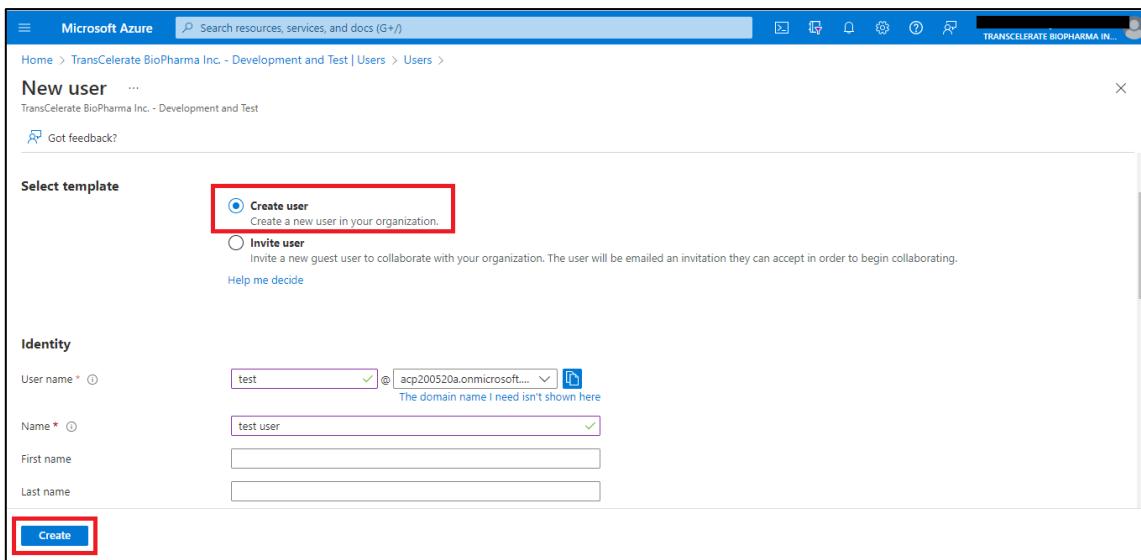
PRE-REQUISITES:

- Global administrator/User Administrator level of access at Active Directory Level.

STEPS:

- i. Login to Azure portal
- ii. Search for Azure Active Directory (AAD)
- iii. Click on the users on the left panel in AAD.
- iv. Click on New User tab on the top, add the user and save the changes.

Figure 7 Create New User



2.4. Role Based Access Controls (RBAC)

GOAL:

Providing access and assigning roles to Azure AD groups for them to access the resources.

PRE-REQUISITES:

- Contributor and User Administrator level of access at Subscription and Active Directory Level respectively.

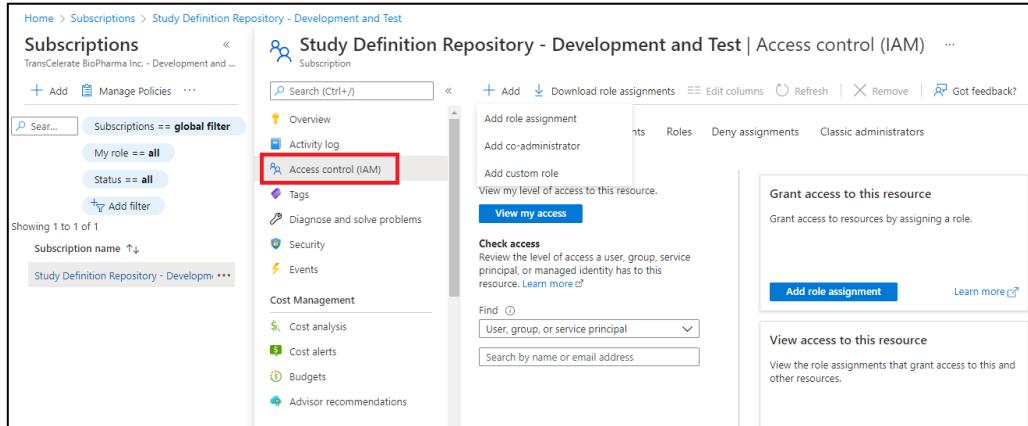
STEPS:

Access Control (IAM) is to limit access and assign roles to groups at the subscription, resource group, and resource level.

At subscription level

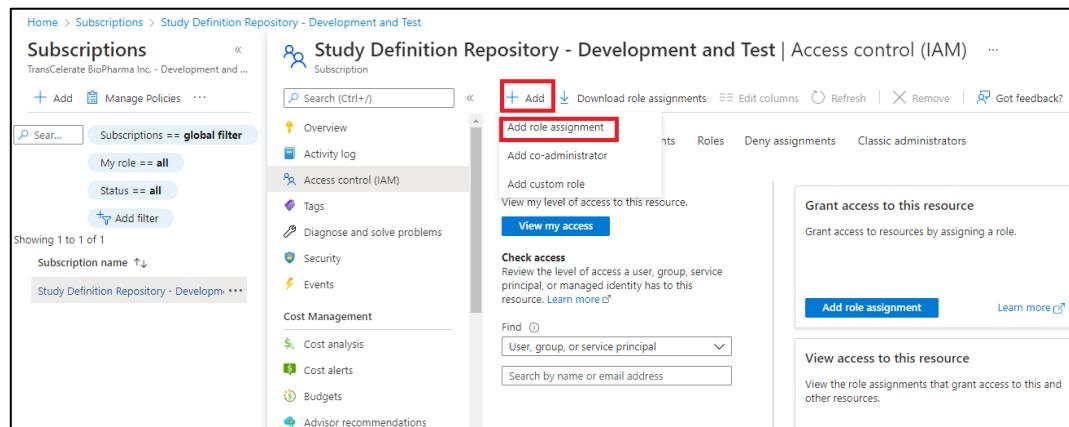
- i. Go to the subscription.
- ii. On the left pane select Access control (IAM) as shown in below screenshot.

Figure 8 Access Control (IAM)



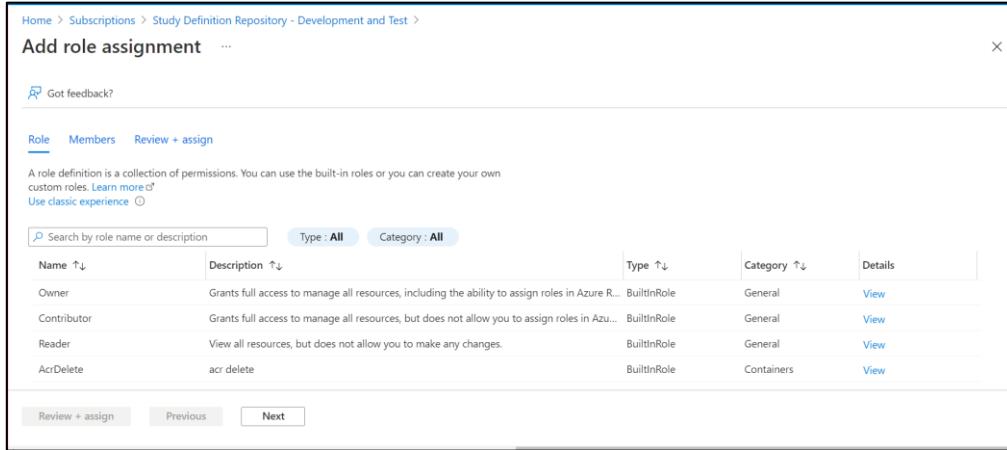
- iii. Click on + Add and add the role assignment

Figure 9 Add Role Assignment



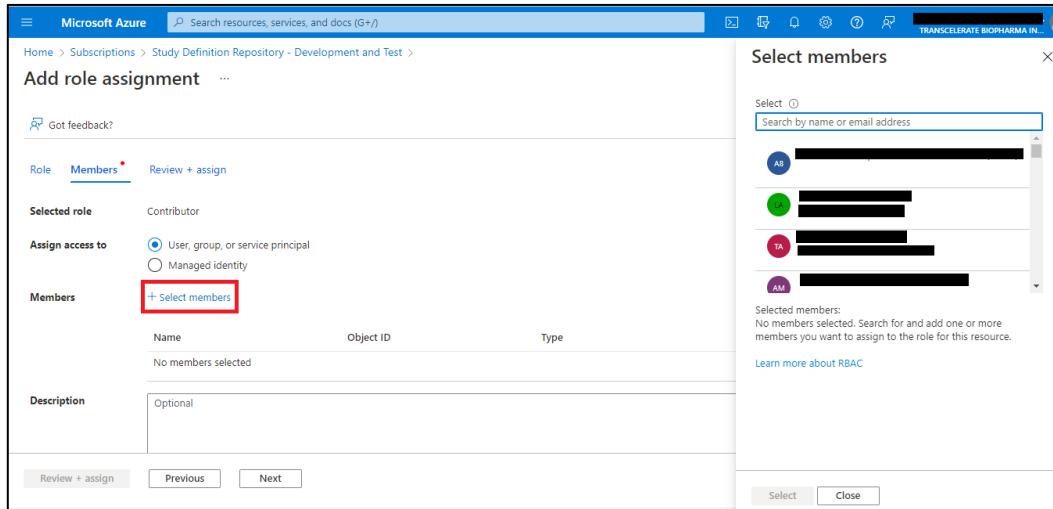
- iv. Select the required role (ex: reader, contributor etc., Refer Table 1) for the members and assign the role to the created groups as shown in the screenshot below and save the changes.

Figure 10 Select Roles



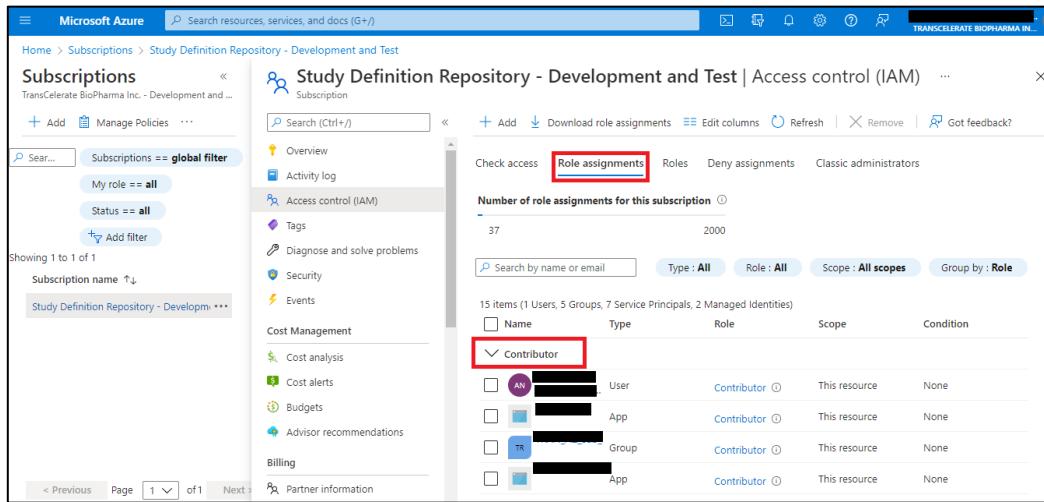
Name	Description	Type	Category	Details
Owner	Grants full access to manage all resources, including the ability to assign roles in Azure R...	BuiltinRole	General	View
Contributor	Grants full access to manage all resources, but does not allow you to assign roles in Aze...	BuiltinRole	General	View
Reader	View all resources, but does not allow you to make any changes.	BuiltinRole	General	View
AcDelete	ac delete	BuiltinRole	Containers	View

Figure 11 Select Members for Role Assignment



- v. To view the roles assigned to a particular group navigate to Role assignments tabs as shown below:

Figure 12 View Role Assignments



The screenshot shows the Microsoft Azure Subscriptions interface. On the left, there's a sidebar with filters like 'Subscriptions == global filter' (My role == all, Status == all). The main area is titled 'Study Definition Repository - Development and Test | Access control (IAM)'. It shows 'Number of role assignments for this subscription' (37) and a table of role assignments. The 'Role assignments' tab is selected. The table has columns: Name, Type, Role, Scope, and Condition. One row is expanded to show details for a 'Contributor' role assigned to a user named 'AN [REDACTED]'. The table shows 15 items (1 Users, 5 Groups, 7 Service Principals, 2 Managed Identities).

Name	Type	Role	Scope	Condition
AN [REDACTED]	User	Contributor	This resource	None
[REDACTED]	App	Contributor	This resource	None
[REDACTED]	Group	Contributor	This resource	None
[REDACTED]	App	Contributor	This resource	None

The same procedure should be followed to provide access/assign roles to any resource in the Azure portal.

2.5. Storage Account and Service Principal Configuration in Azure

GOAL:

Setup storage account and service principal in Azure to enable deployment from GitHub.

PRE-REQUISITES:

- Contributor level of access at Active Directory Level.

STORING THE TERRAFORM STATE FILE REMOTELY:

When deploying resources with Terraform, a state file must be stored; this file is used by Terraform to map Azure Resources to the configuration that you want to deploy, keeps track of meta data, and can also help with improving performance for larger Azure Resource deployments.

- Create Storage Account and Blob Container for storing State file remotely.
- Perform the below commands on Azure CLI for storage Account creation.

Table 2 Azure CLI Code Snippet - Create Storage Account

```
# Create Resource Group
az group create -n ResourceGroupName -l eastus2

# Create Storage Account
az storage account create -n StorageAccountName -g ResourceGroupName -l
eastus2 --sku Standard_LRS
```

```
# Create Storage Account Container
az storage container create -n StorageBlobContainerName --account-name
StorageAccountName --auth-mode login
```

AZURE SERVICE PRINCIPAL:

Create a service principal that will be used by Terraform to authenticate to Azure and assign role to this newly created service principal (RBAC) to the required subscription.

- i. Perform the below command on Azure CLI and capture the JSON output and create an AZURE_SP secret on GitHub and provide the captured output as value for the secret.
- ii. Provide User Administrator access on Azure AD and User Access administrator access on Azure Subscription.

Table 3 Azure CLI Code Snippet - Create Azure Service Principal

```
# Create Service Principal

az ad sp create-for-rbac --name "ServicePrincipal" --role contributor -
--scopes /subscriptions/[enter subscription id] --sdk-auth

Service Principal Sample JSON output:
{
    "clientId": "***Client-Id***",
    "clientSecret": "***Client-Secret***",
    "subscriptionId": "***SusbscriptionId***",
    "tenantId": "***TenantID***",
    "activeDirectoryEndpointUrl":
        "https://login.microsoftonline.com",
    "resourceManagerEndpointUrl": "https://management.azure.com/",
    "activeDirectoryGraphResourceId": "https://graph.windows.net/",
    "sqlManagementEndpointUrl":
        "https://management.core.windows.net:8443/",
    "galleryEndpointUrl": "https://gallery.azure.com/",
    "managementEndpointUrl": "https://management.core.windows.net/"
}
```

2.6. Adding Secrets in GitHub

GOAL:

Configure secrets in GitHub used by GitHub Actions during deployment workflow execution.

PRE-REQUISITES:

- Repo Admin level of access on GitHub Repository
- Capture below secret values based on environment design decisions and storage account, service principal details from Section 2.5

Table 4 GitHub Secrets

Secret Name	Values
AZURE_SP	The Service Principal details in JSON format.
AZURE_RESOURCEGROUP	The name of the Resource Group that contains the storage account.
AZURE_STORAGEACCOUNT	The Storage Account name
AZURE_CONTAINERNAME	The name of the blob container wherein the Terraform State file will be stored.
AZURE_CLIENT_ID	The Client Id of the service principal.
AZURE_CLIENT_SECRET	The Client Secret value of the service principal.
AZURE_SUBSCRIPTION_ID	The Azure Subscription ID
AZURE_TENANT_ID	The Azure Tenant ID
AZURE_RMKEY	Terraform state file name for each environment. (Eg: xxxxdev.tfstate).
Env	Provide the name of the environment (for example, Dev or QA), which will be added to the resource naming convention.
VNet-IP	Provide the VNet Address Space
Subnet-IP	Provide the Subnet Address Space
Subnet-Dsaddress1	Provide the Delegated Subnet1 Address Space
Subnet-Dsaddress2	Provide the Delegated Subnet2 Address Space
Subnet-Dsaddress3	Provide the Delegated Subnet3 Address Space
subscription	Provide the short form of the subscription name; this will be added to the resource naming convention.
Publisher-Name	Provide the Publisher name for API Management Resource.
Publisher-Email	Provide the publisher email id for API Management Resource.
ADgroup1	Provide the name of the Azure AD Group for contributor access to the App Resource Group (Admin Group).
ADgroup2	Provide the name of the Azure AD Group for contributor access to the App Resource Group (DevelopmentTeam_Group).

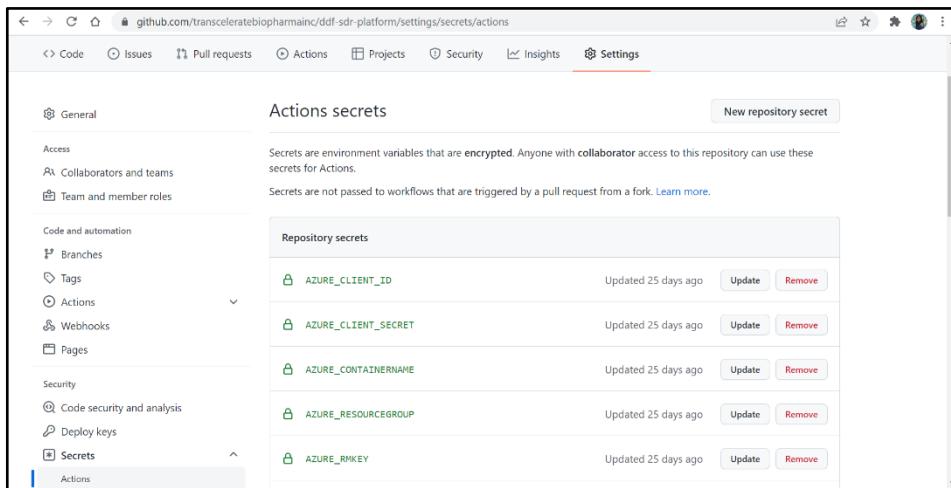
Secret Name	Values
ADgroup3	Provide the name of the Azure AD Group for Reader access on App & Core Resource Groups.
Serviceprincipal	Provide the name of the Service principal that was created for the Git connection; it will provide key vault secret user access and access policies for secrets on Key Vault for the Service Principal.

STEPS:

Add GitHub Secrets entries for all the secrets captured in the pre-requisites.

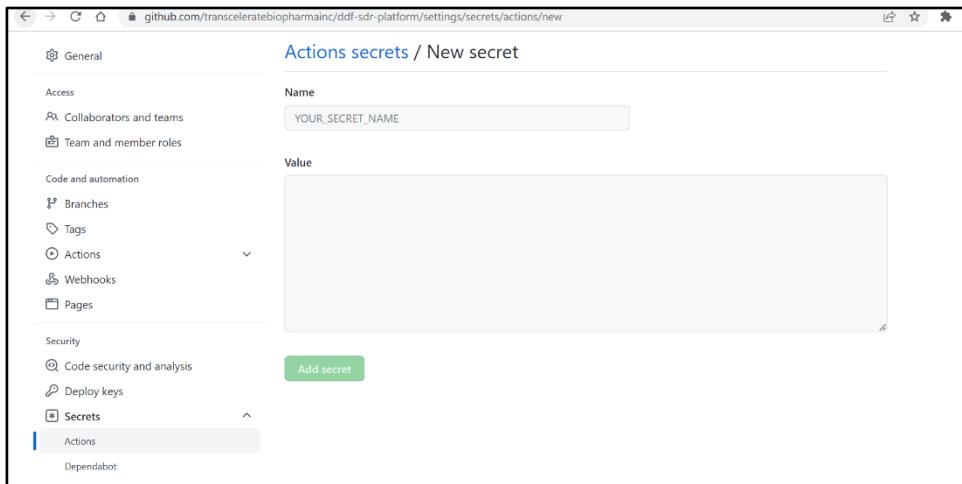
- i. Go to repository settings.
- ii. On the lower left-hand side of the screen, click on Secrets.
- iii. Under that click on Actions.
- iv. Then click on New Repository Secret.

Figure 13 GitHub Actions Secrets



- v. After clicking New Repository Secret, fill the details of the secret (name and value) in the boxes

Figure 14 Add new GitHub Action Secret



2.7. Execute GitHub Action for IaC deployment

GOAL:

Execute the GitHub actions to deploy the Azure resources using IaC code.

PRE-REQUISITES:

- Repo Admin level of access on GitHub Repository
- For setting up the actions in GitHub, user must have Write permission on repos.

DEPLOYMENT:

The folder “.github/workflows” in the IaC Repository on GitHub contains the GitHub Actions yaml script (main.yml) for deploying the Terraform IaC code on Microsoft Azure Platform. Note that, the deployment scripts are specific to each release included with release tags on SDR GitHub code repositories.

main.yml:

The yaml file is a multi-job script that will perform security checks on IaC code as well as the deployment of resources to the target environment on Microsoft Azure Platform.

STEPS:

- i. Go to GitHub Actions and under the list of workflows click on CI.
- ii. In this workflow click Run Workflow to trigger the Deployment Action.
- iii. Once the workflow completes successfully, the SDR Solution resources should have been deployed to Azure platform.

2.8. Manual Configuration Changes on Azure Platform

2.8.1. OAuth 2.0 Configuration for SDR UI Application

GOAL:

Azure AD Client Application Registration and OAuth 2.0 configuration for SDR UI application.

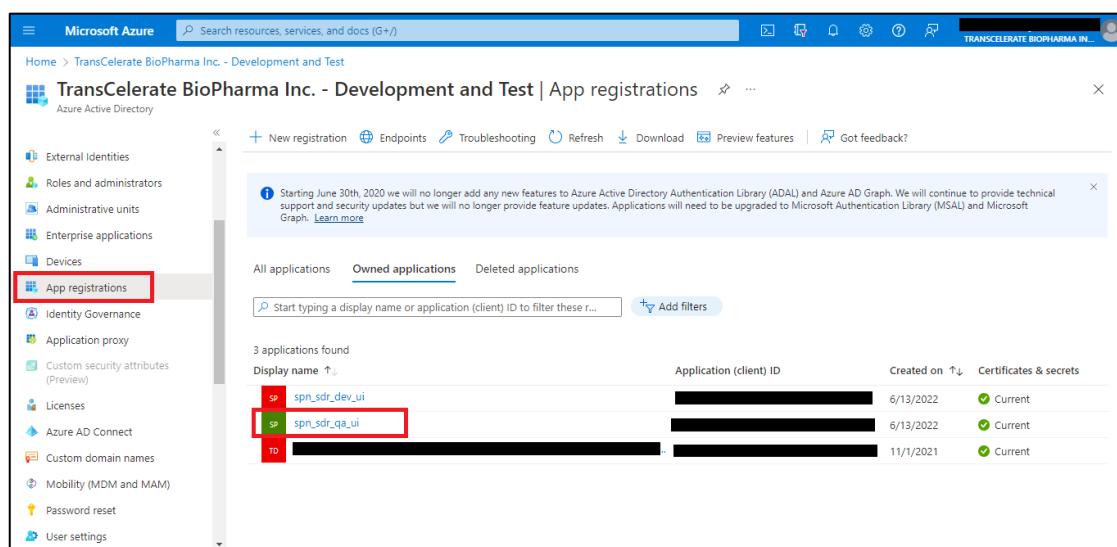
PRE-REQUISITES:

- Global Administrator level of access at Active Directory level.

STEPS**SDR UI APP REGISTRATION:**

- i. Go To Azure AD → App Registrations → Select UI App Registration

Figure 15 Azure AD - App Registration

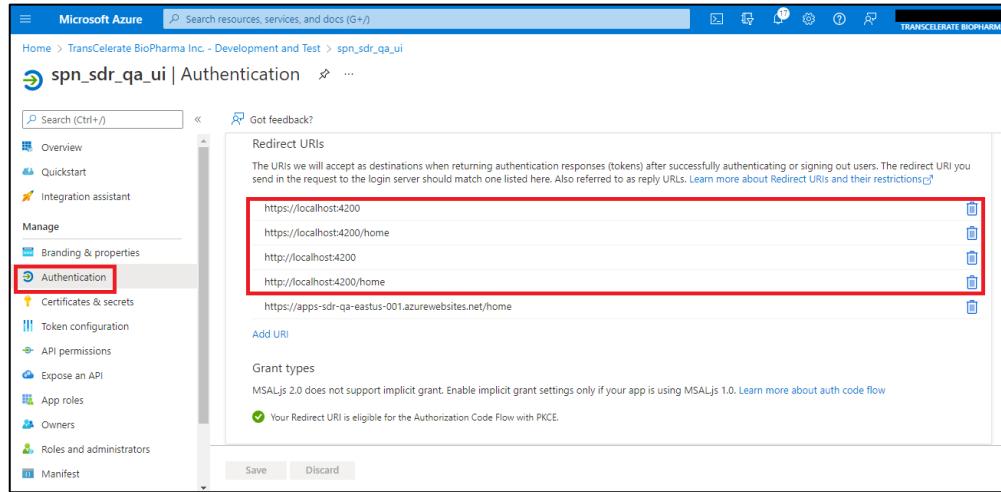


The screenshot shows the Microsoft Azure portal interface for 'App registrations'. The left sidebar has a red box around the 'App registrations' option under 'Azure Active Directory'. The main area shows a table of registered applications. Two entries are highlighted with red boxes: 'spn_sdr_dev_ui' and 'spn_sdr_qa_ui'. The table columns include 'Display name', 'Application (client) ID', 'Created on', and 'Certificates & secrets'. A message at the top states: 'Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure AD Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph.' [Learn more](#)

Display name	Application (client) ID	Created on	Certificates & secrets
spn_sdr_dev_ui	[REDACTED]	6/13/2022	Current
spn_sdr_qa_ui	[REDACTED]	6/13/2022	Current
TO [REDACTED]	[REDACTED]	11/1/2021	Current

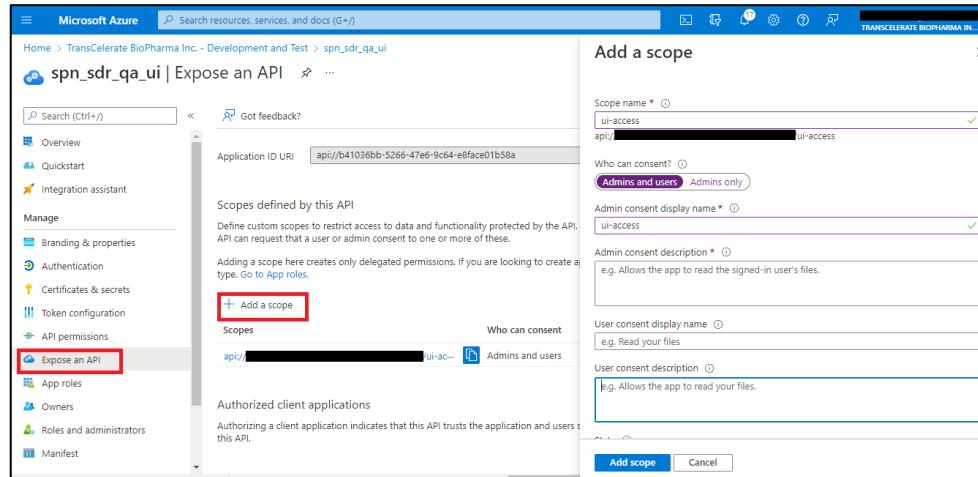
- ii. Go to Authentication blade, add additional Redirect URL as needed. Add localhost URL's for testing the application from development machine.

Figure 16 Azure AD - App Registration - Redirect URI



- iii. Go to “Expose an API” blade and click on “Add a Scope”. Provide scope name, select “Admins and users” in “Who can consent”, provide admin consent display name and finally click on “Add Scope”.

Figure 17 Azure AD - App Registration - Expose an API



- iv. Go to API Permissions → Click on Add a permission → Select My Api's → Select Backend app registration exposed Api on step 4. → select Api (ui-access) → click Add Permission.

Figure 18 Azure AD - App Registration - API Permission

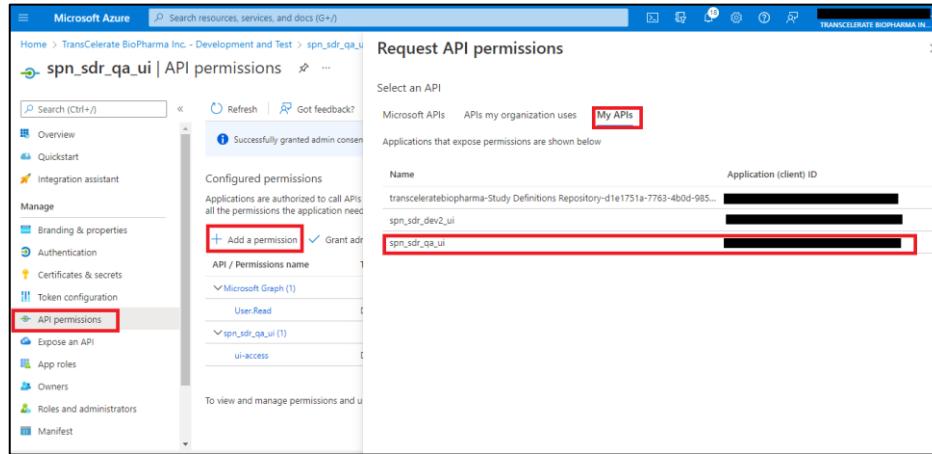
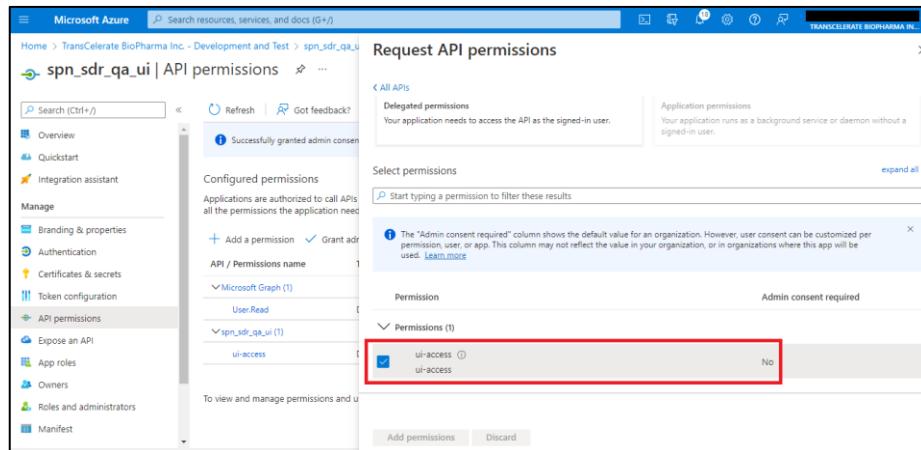
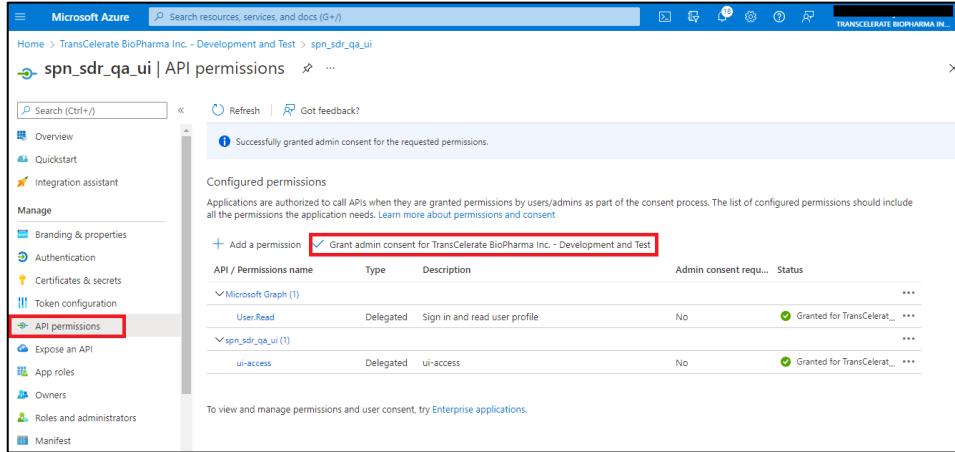


Figure 19 Azure AD - App Registration - Add API Permission



v. Grant Admin consent.

Figure 20 Azure AD - App Registration - API Permission - Grant Admin Consent

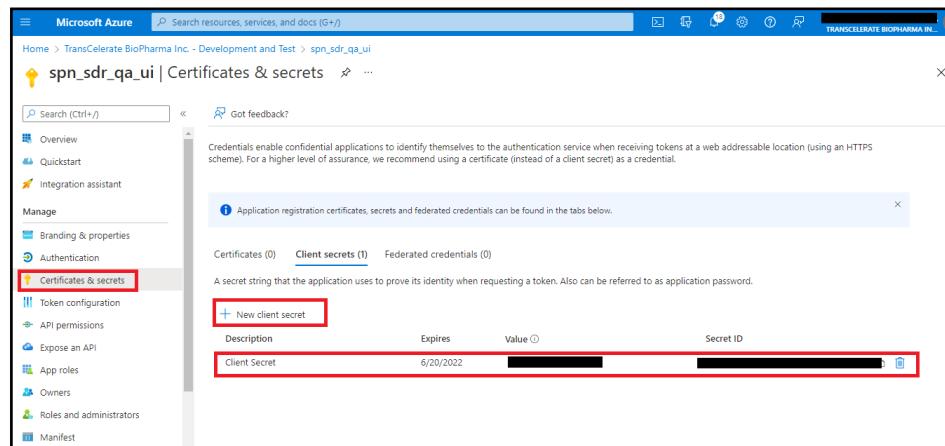


The screenshot shows the Microsoft Azure portal interface for app registration. The left sidebar has 'API permissions' selected. A message at the top says 'Successfully granted admin consent for the requested permissions.' Below it, under 'Configured permissions', there is a table:

API / Permissions name	Type	Description	Admin consent req...	Status
User.Read	Delegated	Sign in and read user profile	No	Granted for TransCelerate BioPharma Inc.
ui-access	Delegated	ui-access	No	Granted for TransCelerate BioPharma Inc.

- vi. Go to certificates & secrets → click on client secrets → click new client secret and copy the value and add it on Key Vault secrets.

Figure 21 Azure AD - App Registration - Certificates & Secrets

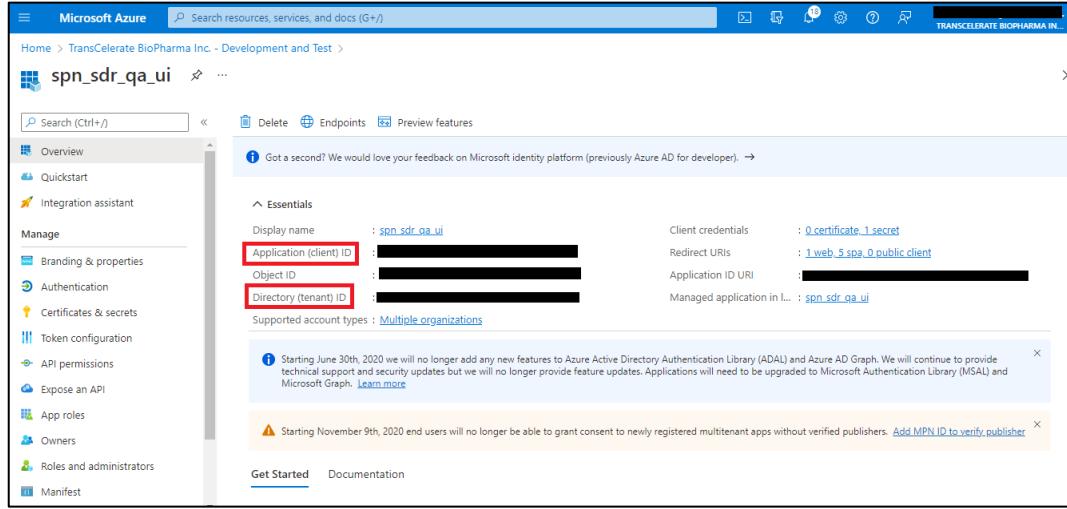


The screenshot shows the Microsoft Azure portal interface for app registration. The left sidebar has 'Certificates & secrets' selected. A message at the top says 'Application registration certificates, secrets and federated credentials can be found in the tabs below.' Below it, under 'Client secrets (1)', there is a table:

Description	Expires	Value	Secret ID
Client Secret	6/20/2022	[REDACTED]	[REDACTED]

- vii. Capture the client id and tenant id and add it on Key Vault secrets. We must generate tokens for authenticating to the SDR UI Application using the client app registration details (client ID, Tenant ID, and Secret value). Refer Section 2.8.3 for Key Vault secrets.

Figure 22 Azure AD - App Registration - Essential Secrets



The Azure AD App registration for UI has been created successfully.

Note: All the users added at AD level will have access to the SDR UI Application by default.

2.8.2. Key Vault

GOAL:

- Add access policy and enable Azure Key Vault Administrator User to manage secrets.
- To create secrets in Azure Key Vault

PRE-REQUISITES:

- Contributor level of access for Resource Group.
- Capture below secret values from the deployed resources.

Table 5 KeyVault Secrets

Secret Name	Secret Value
Apim-BaseUrl	Provide API Management URL as key-value (E.g., https://apim-sdr-qa-eastus.azure-api.net/studydefinitionrepository/v1/)
ApplicationInsights--InstrumentationKey	Provide Application Insights Instrumentation key
AzureAd-Audience	Provide the Azure App Registration Scope URL, Refer to Section 2.8.1 step 3 for scope URL (E.g.: api://0000-0000-000-000/ui-access)

Secret Name	Secret Value
AzureAd--Audience	Provide the UI app registration Application ID URI (E.g.: api://0000-0000-0000-0000)
AzureAd-ClientSecret	Provide App Registration Client secret value.
AppInsights-ApiKey	Provide Application Insights Api Key Value
AppInsights-AppId	Provide Application Insights AppId
AppInsights-RESTApiUrl	Provide Default URL https://api.applicationinsights.io/v1/apps
AzureAd-Authority	Provide the Azure Ad authority value (https://login.microsoftonline.com/ (Provide Azure AD Tenant ID))
AzureAd-ClientId	Provide the App Registration client ID, refer to Section 2.8.1 App Registration step 7 for client ID
AzureAd-LoginUrl	Provide the Front End (UI) App Service URL.
AzureAd-RedirectUrl	Provide the Redirect URL (E.g.: https://Front End App service URL (UI)/home)
AzureAd-TenantId	Provide the Azure AD Tenant ID, refer to Section 2.8.1 App Registration step 7 for Tenant ID
ConnectionStrings--DatabaseName	Provide the Cosmos DB Database Name.
ConnectionStrings--ServerName	Provide the Cosmos DB connection string.
Azure-SP	Store the Azure Service Principal JSON to utilize for API and UI deployments.
isAuthEnabled	true
isGroupFilterEnabled	true
StudyHistory--DateRange	30
Env-Name	Provide a key word for the deployed environment (dev,qa,stage etc.)
ApiVersionUsdmVersionMapping	Copy JSON content from file : ddf-sdr-api-usdm-version-mapping
SdrCptMasterDataMapping	Copy JSON content from file : ddf-sdr-cpt-master-data-mapping
ConformanceRules	Copy JSON content from file : ddf-sdr-usdm-conformance-rules-configuration
AzureServiceBusConnectionString	Provide the Azure Service Bus Connection String

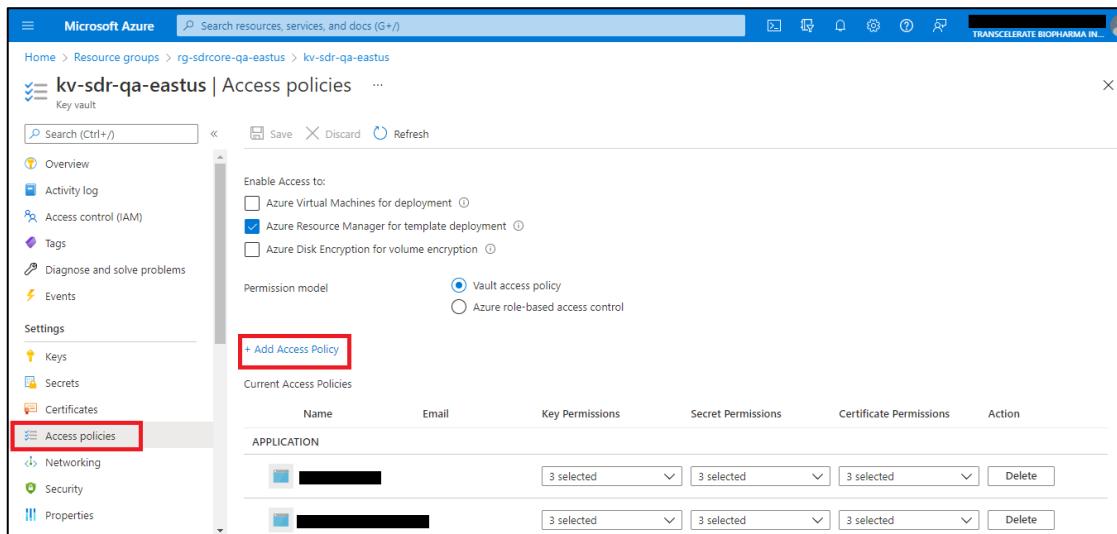
Secret Name	Secret Value
AzureServiceBusQueueName	Provide the Azure Service Bus Queue Name

STEPS FOR ADDING ACCESS POLICY:

The steps for adding Key Vault Administrator to Key Vault access policies for creating secrets are listed below.

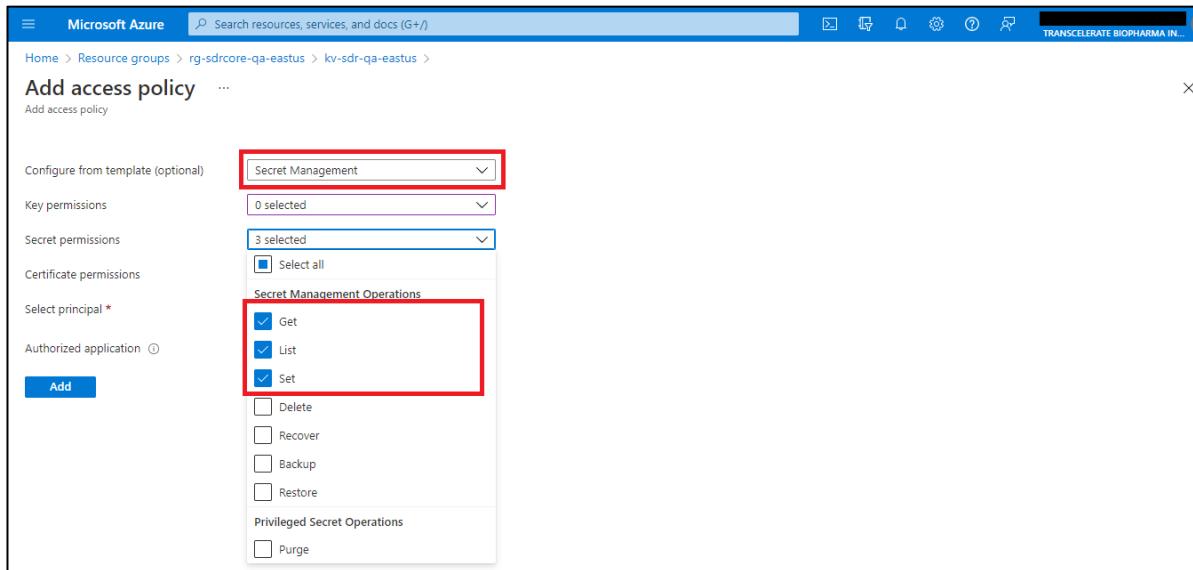
- i. Go to → Key Vault → Select Access Policies → Click on Add Access Policy

Figure 23 Add Key Vault Access Policy



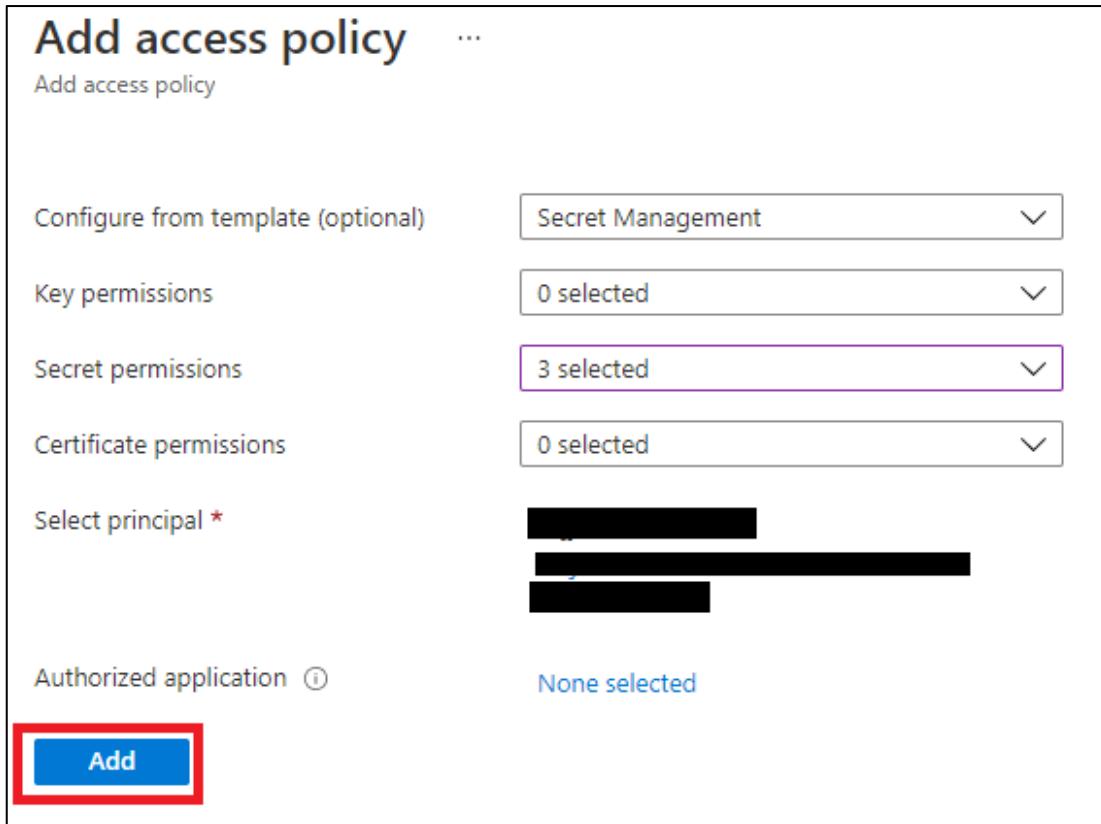
- ii. Select Secret Management → select Secret Permissions (Get, List, Set)

Figure 24 Select Secret Permissions in Access Policy in Key Vault



- iii. Select Principal → Add Azure Key Vault Administrator User (select the username) → Click Add

Figure 25 Select Principal (List of users) In Key Vault Access Policy



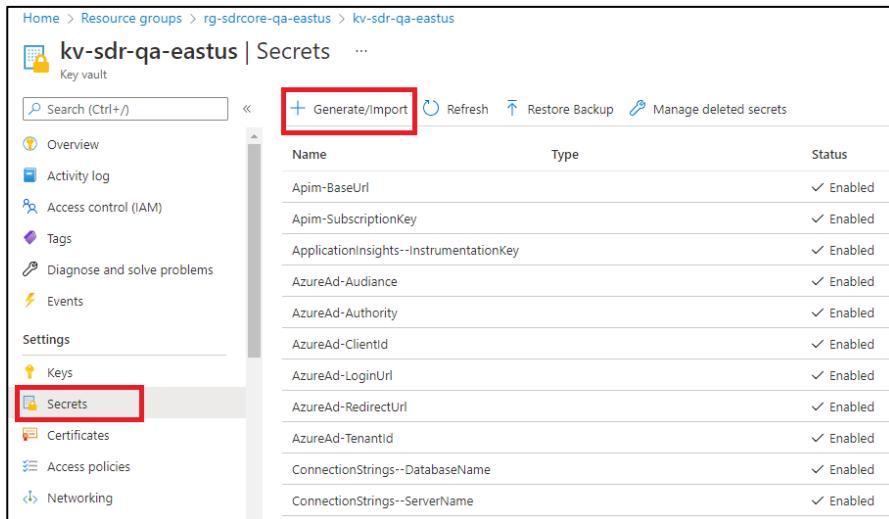
The screenshot shows the 'Add access policy' dialog. In the 'Select principal *' section, there is a list of user names, with three names highlighted with a black redaction box. Below this, the 'Authorized application' section shows 'None selected'. The 'Add' button at the bottom left is highlighted with a red box.

STEPS FOR ADDING KEY VAULT SECRETS:

Add Key Vault entries for all the secrets captured in the pre-requisites.

- i. Go to → Key Vault → Select Secrets → Click Generate/Import

Figure 26 Create Secrets in Key Vault

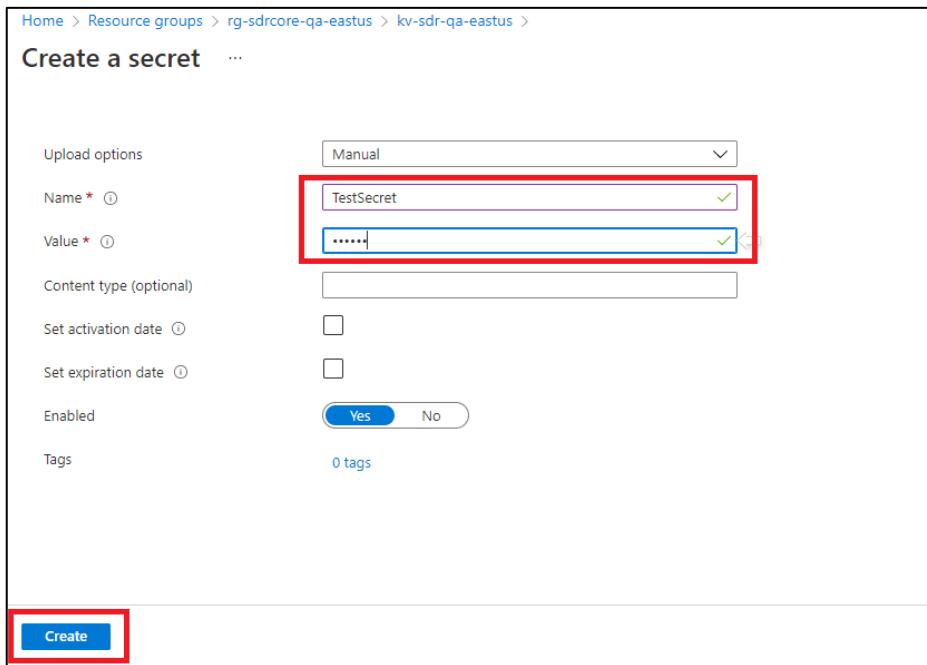


The screenshot shows the 'kv-sdr-qa-eastus' Key Vault blade in the Azure portal. The left sidebar has 'Secrets' selected. The main area lists various secrets with their names, types, and status. A red box highlights the '+ Generate/Import' button at the top right of the list.

Name	Type	Status
Apim-BaseUrl		✓ Enabled
Apim-SubscriptionKey		✓ Enabled
ApplicationInsights--InstrumentationKey		✓ Enabled
AzureAd-Audience		✓ Enabled
AzureAd-Authority		✓ Enabled
AzureAd-ClientId		✓ Enabled
AzureAd-LoginUrl		✓ Enabled
AzureAd-RedirectUrl		✓ Enabled
AzureAd-TenantId		✓ Enabled
ConnectionString--DatabaseName		✓ Enabled
ConnectionString--ServerName		✓ Enabled

- ii. Provide Secret Name and Value → Select Create

Figure 27 Add Secrets values in Key Vault



The screenshot shows the 'Create a secret' dialog in the Azure portal. It includes fields for 'Name' (TestSecret), 'Value' (redacted), 'Content type (optional)', 'Set activation date', 'Set expiration date', 'Enabled' (Yes), and 'Tags'. A red box highlights the 'Create' button at the bottom.

2.8.3. Storage Account

GOAL:

- Disable public Access for Blob storage

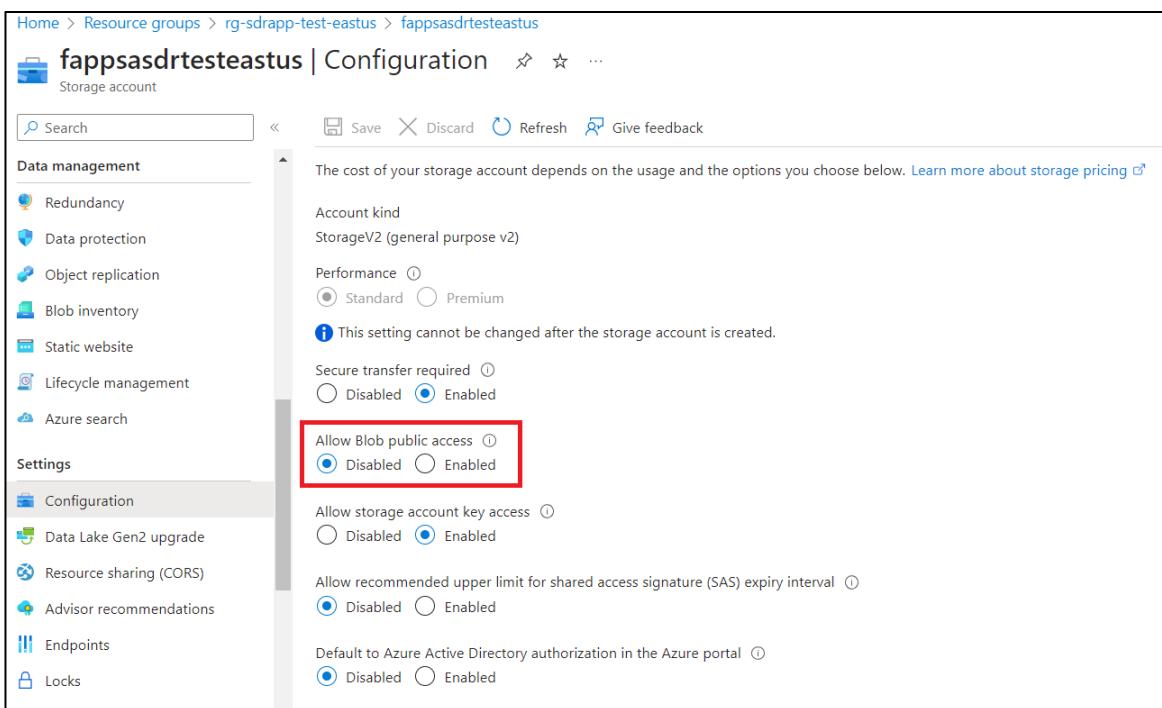
PRE-REQUISITES:

- Contributor level of access for Resource Group.

STEPS TO DISABLE PUBLIC ACCESS FOR STORAGE ACCOUNT:

- i. Go to → Storage account → Select Configuration → Click on Disabled (Allow Blob public access) → Save

Figure 28 Disable public access for Blob storage



3. Resource Validation

Validate all resources from Azure Portal to ensure that the resource configurations have been deployed in accordance with the [low-level design document \(LLD\)](#).

3.1. Low-Level Design Document

The low-level design document contains all the configurations and settings of Azure resources required for SDR Reference Implementation Platform that have been configured on Terraform IaC code.

Naming Convention followed for all the resources is as below -

- Resource type: *vnet, subnet, rg, etc.*
- App/Svc: *Subscription name*
- Environment: *dev, preprod etc.*
- Region: *eastus, westus, etc.*

Figure 29 Resource Naming Convention

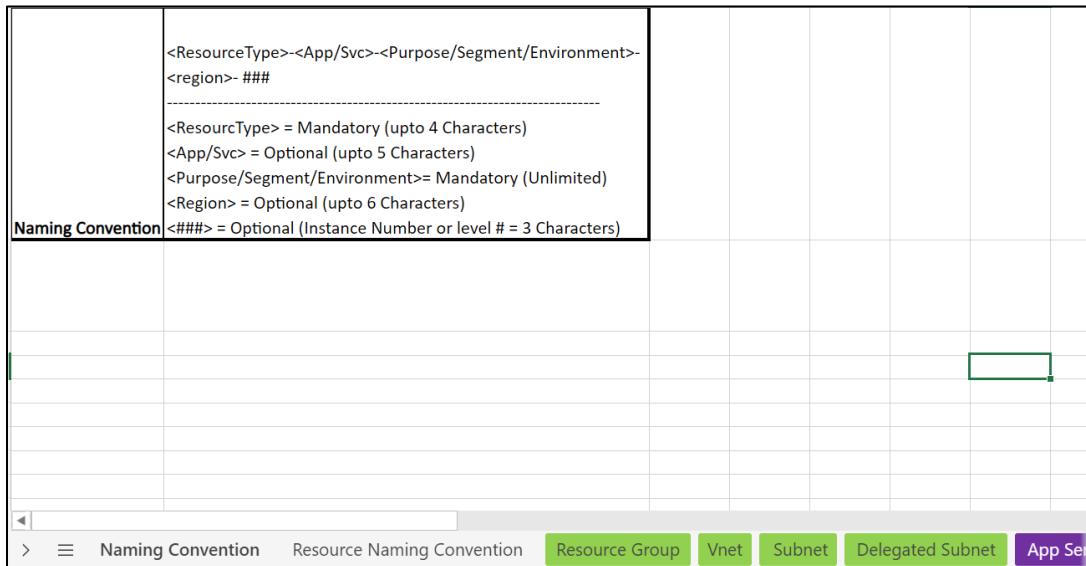


Figure 30 SDR Resource Naming Convention

A	B
1 Resource Naming Convention	Environment (Eg: Dev)
2 Resource Group1	rg-SubNamecore-EnvName-eastus
3 Resource Group2	rg-SubNameapp-EnvName-eastus
4 Vnet	vnet-SubName-EnvName-eastus
5 Subnet	snet-SubName-EnvName-eastus
6 Delegated Subnet1	dsnet-SubNameui-EnvName-eastus-001
7 Delegated Subnet2	dsnet-SubNameapi-EnvName-eastus-002
8 App Service1	apps-SubNameui-EnvName-eastus-001
9 App Service2	apps-SubNameapi-EnvName-eastus-002
10 App Service Plan1	asp-SubNameui-EnvName-eastus-001
11 App Service Plan2	asp-SubNameapi-EnvName-eastus-002
12 API Management	apim-SubName-EnvName-eastus
13 Application Insights	appin-SubName-EnvName-eastus
14 CosmosDB	cdb-SubName-EnvName-eastus
15 Log Analytics Workspace	law-SubName-EnvName-eastus
16 Key Vault	kv-SubName-EnvName-eastus
17 Storage Account	saSubNameEnvNameeastus
18 Terraform State File	tfstatefileEnvName
19 Diagnostic Setting Name	diags-vnet-SubName-EnvName-eastus
20	
21	
22	

Note: For Resource Naming Convention best practices please refer [Azure Resource Naming Conventions](#)

3.2. Virtual Network

Validation of Virtual Network (VNet) configuration

Figure 31 VNet Configurations

Vnet					EnvName		
Basics	Project details		Subscription				
			Resource group	rg-SubNamecore-EnvName-eastus			
	Instance details		Name	vnet-SubName-EnvName-eastus			
			Region	East US			
IP Addresses	IPv4 address space		xxx.xxx.x.x/24				
Security	BastionHost	Disable	Disable				
			Enable				
	DDoS Protection Standard		Disable	Disable			
			Enable				
	Firewall	Disable	Disable				
			Enable				
Tags	Name1:Value1		Environment: EnvName				
Name2:Value2			App Layer: N/A				
Diagnostic Sett	Diagnostic setting name		diags-vnet-SubName-EnvName-eastus				
	Logs		Category group	allLogs			
			Categories	VMProtection Alerts			
	Metrics		AllMetrics		Enable		
Destination details		Send to Log Analytics Workspace		Enable			
	Resource Naming Convention	Resource Group	Vnet	Subnet	Delegated Subnet		
			API Management	App Service Plan	App		

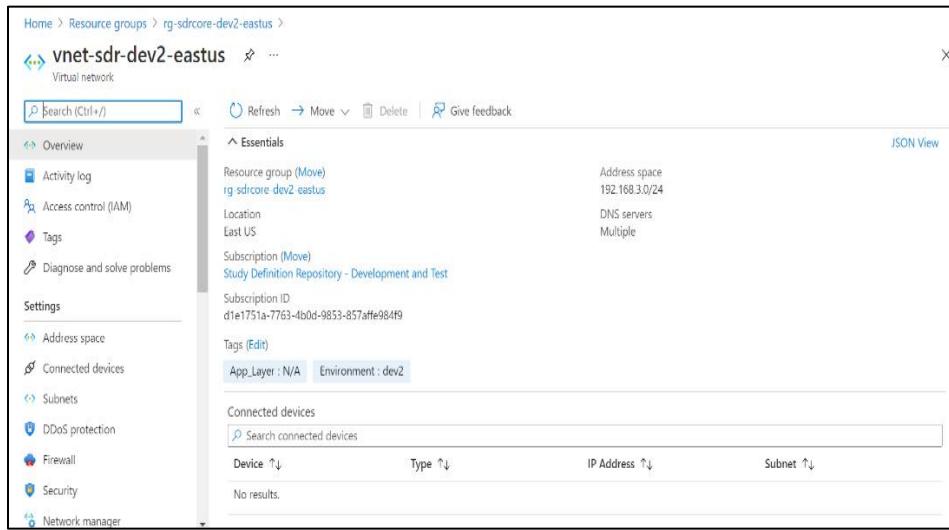
PRE-REQUISITES:

- Reader access at Resource group level
 - SDR Reference Implementation Low Level Design Document (LLD) document

STEPS

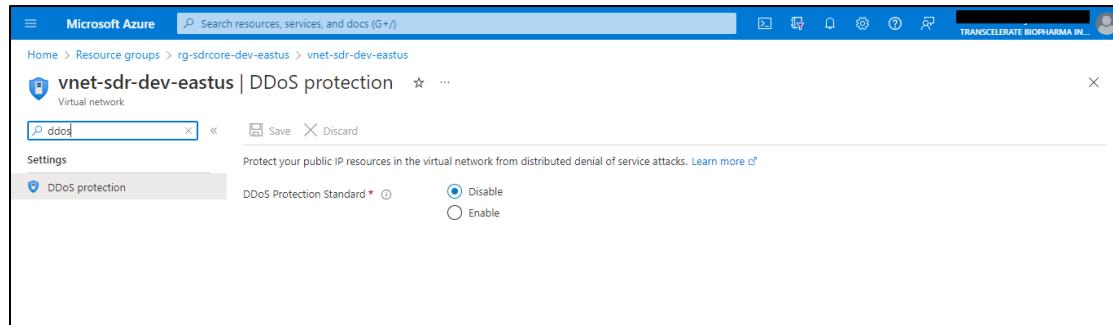
- i. Login to Azure Portal
 - ii. Click on the Resource Groups tab.
 - iii. Select the Resource group for VNet configuration.
 - iv. Verify that the Basic details like Subscription, Resource Group, Name and Region is as per the LLD
 - v. Verify that the IPv4 Address Space is as per the LLD

Figure 32 Virtual Network



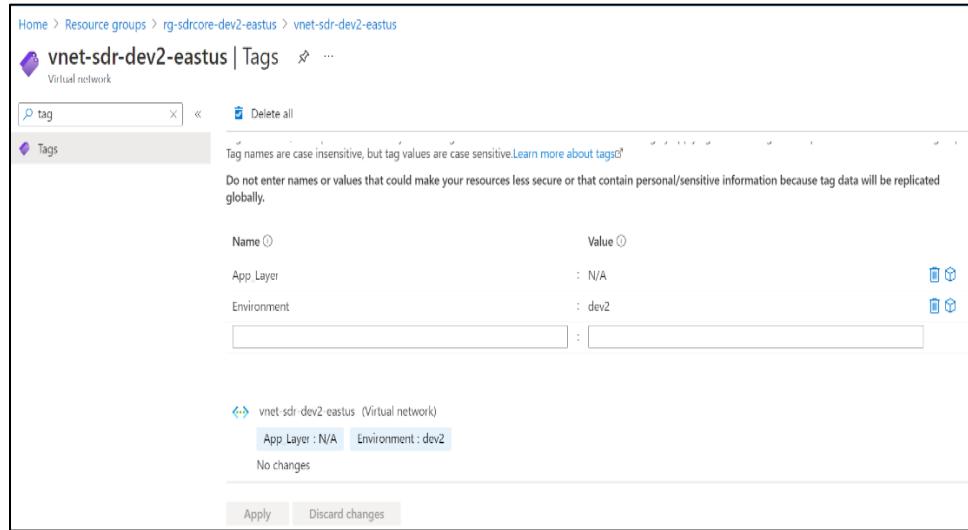
- vi. Go to Security tab and verify that Bastion Host is set as per the LLD.
- vii. Go to DDoS Protection tab and verify that it is set as per the LLD.

Figure 33 Virtual Network - DDoS protection



- viii. Go to Firewall tab and verify that it is set as per the LLD.
- ix. Go to the Tags tab and verify that the Environment and App Layer should be as per the LLD.

Figure 34 Virtual Network - Tags

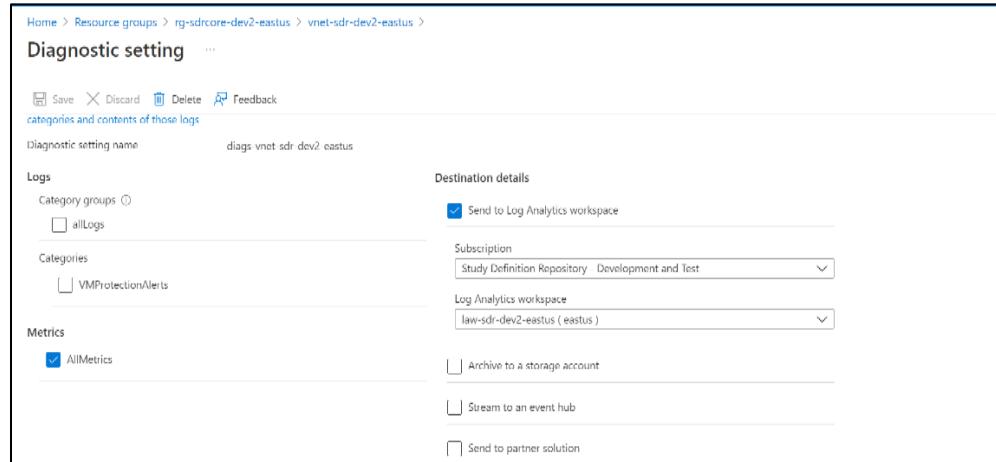


Name	Value
App Layer	N/A
Environment	dev2

x. Go to the Diagnostic Settings Tab and verify that the below settings are as per LLD.

- Diagnostic setting name
- Configuration for
 - Logs
 - VM Protection Alerts
 - All Metrics
 - Send to Log Analytics Workspace
 - Archive to a Storage Account
 - Stream to an Event Hub
 - Send to Partner Solution
- Subscription Name
- Log Analytics Workspace name

Figure 35 Virtual Network - Diagnostic Setting



3.3. Subnet

Validation of Virtual Subnet configuration.

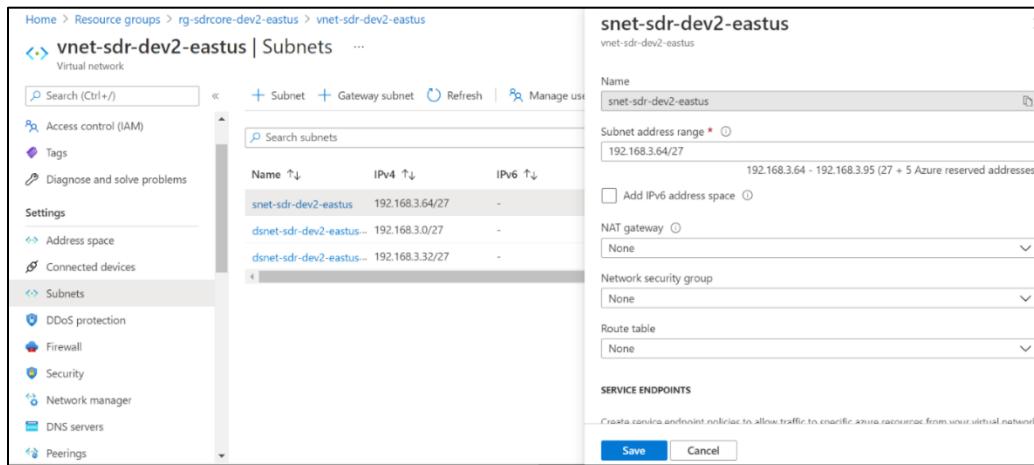
PRE-REQUISITES:

- Reader level of access at Resource group level
- SDR Reference Implementation Low Level Design Document (LLD) document

STEPS:

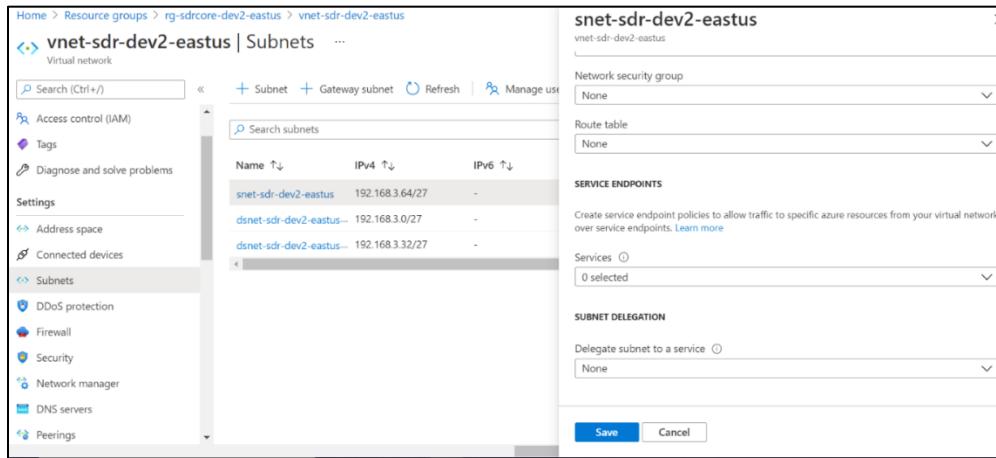
- i. Login to Azure Portal
- ii. Click on the Resource Groups tab.
- iii. Select the Resource group for VNet configuration.
- iv. Verify that the below basic details are as per the LLD.
 - Name
 - Subnet address range
 - Add IPv6 address space.
 - NAT gateway
 - Network Security Group
 - Route table
 - Services
 - Delegate subnet to a service

Figure 36 Subnet Details



The screenshot shows the Azure portal interface for managing a virtual network. On the left, there's a sidebar with various settings like Access control (IAM), Tags, and Subnets. The main area shows a list of subnets under 'vnet-sdr-dev2-eastus'. A new subnet is being created with the name 'snet-sdr-dev2-eastus' and an IPv4 address range of 192.168.3.64/27. The configuration pane on the right includes fields for Name, Subnet address range, NAT gateway (set to None), Network security group (set to None), and Route table (set to None). A note at the bottom suggests creating a service endpoint rule.

Figure 37 Subnet Details



3.4. Delegated Subnet

Validation of Delegated Subnet configuration.

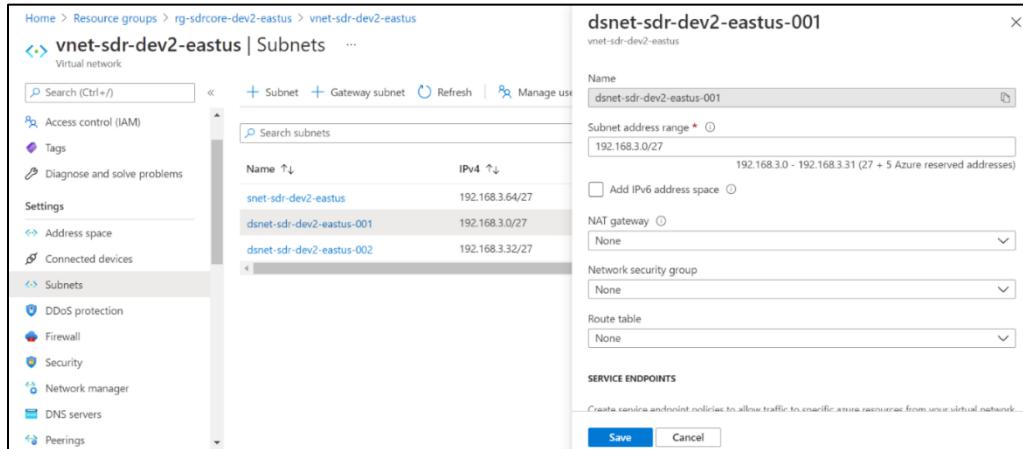
PRE-REQUISITES:

- Reader level of access at Resource group Level.

STEPS

- i. Login to Azure Portal
- ii. Click on the Resource Groups tab.
- iii. Select the Resource group for VNet configuration.
- iv. Verify that the below basic details are as per the LLD.
 - Name
 - Subnet address range
 - Add IPv6 address space.
 - NAT gateway
 - Network Security Group
 - Route table
 - Services
 - Delegate subnet to a service

Figure 38 Delegated Subnet-001 Details



dsnet-sdr-dev2-eastus-001

Name: dsnet-sdr-dev2-eastus-001

Subnet address range: 192.168.3.0/27 (192.168.3.0 - 192.168.3.31 (27 + 5 Azure reserved addresses))

NAT gateway: None

Network security group: None

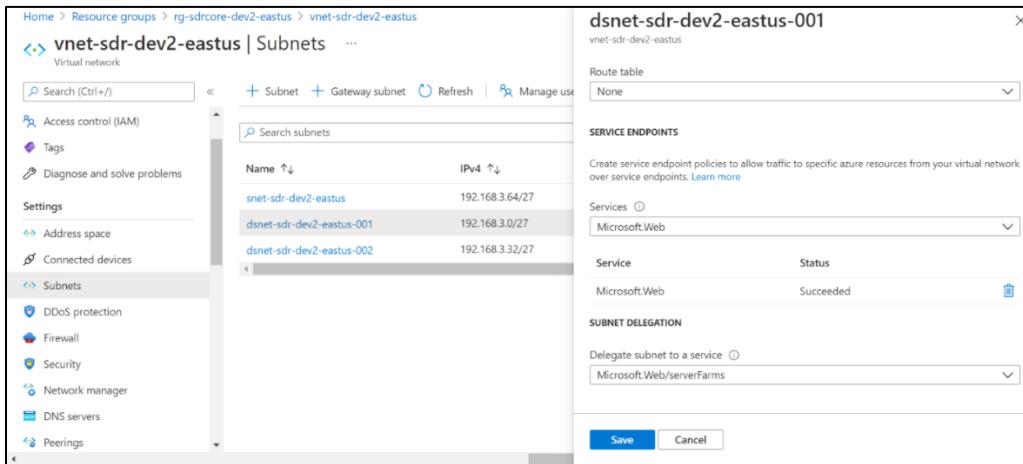
Route table: None

SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific Azure resources from your virtual network. Learn more

Save Cancel

Figure 39 Delegated Subnet-001 Service Endpoints Details



dsnet-sdr-dev2-eastus-001

Route table: None

SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific Azure resources from your virtual network over service endpoints. Learn more

Services: Microsoft.Web

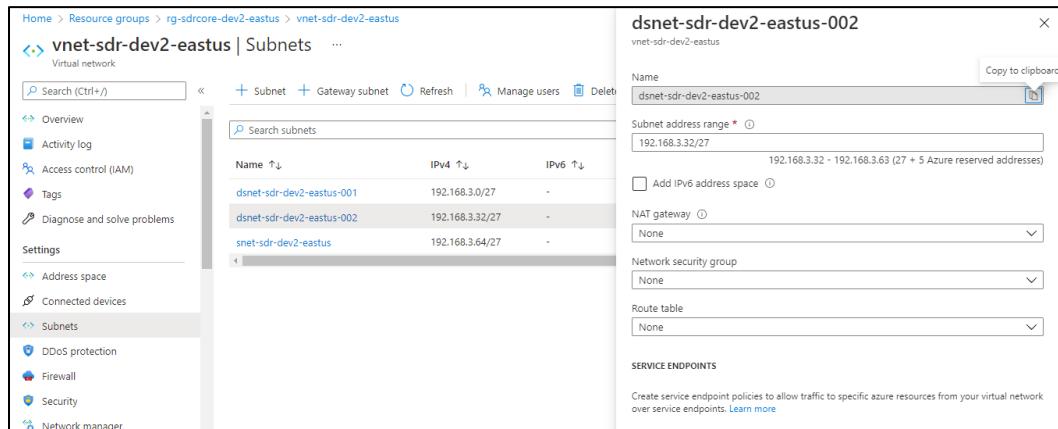
Service	Status
Microsoft.Web	Succeeded

SUBNET DELEGATION

Delegate subnet to a service: Microsoft.Web/serverFarms

Save Cancel

Figure 40 Delegated Subnet-002 Details



dsnet-sdr-dev2-eastus-002

Name: dsnet-sdr-dev2-eastus-002

Subnet address range: 192.168.3.32/27 (192.168.3.32 - 192.168.3.63 (27 + 5 Azure reserved addresses))

NAT gateway: None

Network security group: None

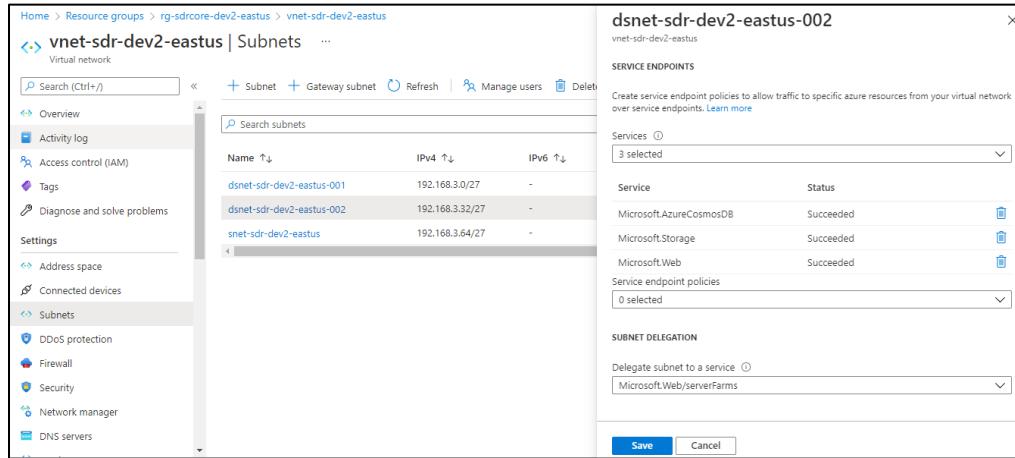
Route table: None

SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific Azure resources from your virtual network over service endpoints. Learn more

Save

Figure 41 Delegated Subnet-002 Service Endpoints Details



dsnet-sdr-dev2-eastus-002

vnet-sdr-dev2-eastus

SERVICE ENDPOINTS

Create service endpoint policies to allow traffic to specific azure resources from your virtual network over service endpoints. [Learn more](#)

Service	Status
Microsoft.AzureCosmosDB	Succeeded
Microsoft.Storage	Succeeded
Microsoft.Web	Succeeded

Service endpoint policies

0 selected

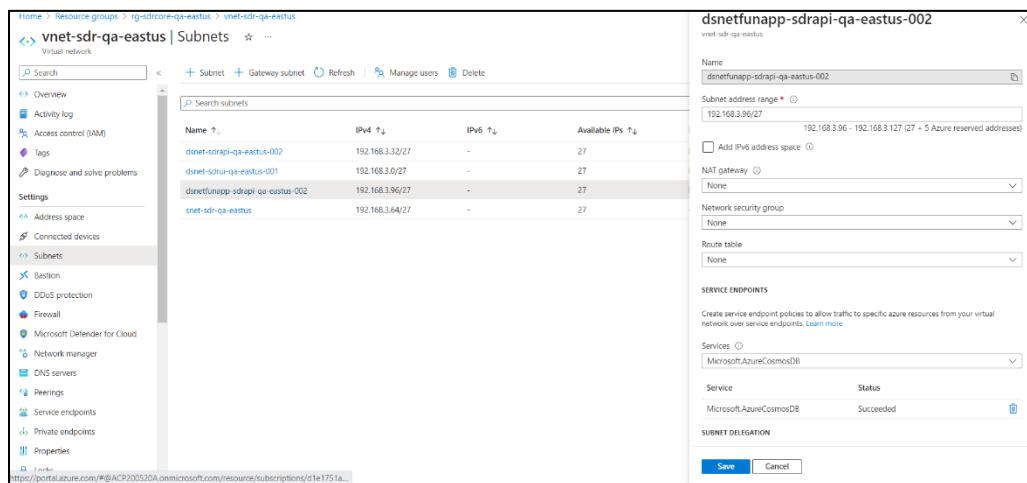
SUBNET DELEGATION

Delegate subnet to a service [\(i\)](#)

Microsoft.Web/serverFarms

Save **Cancel**

Figure 42 Function App Delegated Subnet-002 Details



dsnetsdrapi-qa-eastus-002

vnet-sdr-qa-eastus

Name: dsnetsdrapi-qa-eastus-002

Subnet address range: 192.168.3.90/27 (192.168.3.96 - 192.168.3.127) (27 + 5 Azure reserved addresses)

Add IPv6 address space [\(i\)](#)

NAT gateway: None

Network security group: None

Route table: None

SERVICE ENDPOINTS

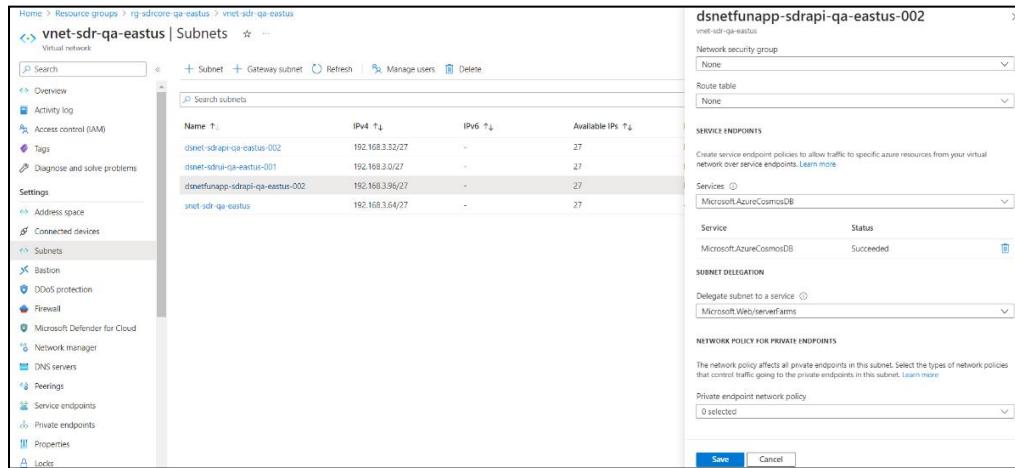
Create service endpoint policies to allow traffic to specific azure resources from your virtual network over service endpoints. [Learn more](#)

Service	Status
Microsoft.AzureCosmosDB	Succeeded

SUBNET DELEGATION

Save **Cancel**

Figure 43 Function App Delegated Subnet-002 Service Endpoints Details



3.5. Other Resources

The similar steps mentioned in previous sections for VNet & Subnet resources verifications should be followed to ensure that all the below resources deployed in the Azure Platform are configured in accordance with the LLD.

- Resource Groups
- App Services
- App Service Plans
- API Management
- Application Insights
- Cosmos DB
- Log Analytics Workspace
- Key Vault
- Azure Service Bus
- Azure Function App
- Azure Container Registry

4. Application Code Deployment

SDR application code from the main branch (which always corresponds to latest SDR release) contains the latest deployment script (main.yml) for both API and UI. Note that, the deployment scripts are specific to each release included with release tags on SDR GitHub code repositories.

SDR Release	Release Tag
SDR Release V0.5	ddf-sdr-ui-release-v0.5 ddf-sdr-api-release-v0.5 ddf-sdr-platform-release-v0.5

SDR Release V2.0	ddf-sdr-ui-release-v2.0 ddf-sdr-api-release-v2.0 ddf-sdr-platform-release-v2.0
------------------	--

4.1 GitHub Secrets used in Workflow

PRE-REQUISITES

- Read access to fetch Key Vault secrets in Azure Portal.
- User Should have access policies set to read/retrieve secrets from Azure Key Vault in Azure Portal.
- User Should have Repo Admin level of access to add/replace the GitHub secrets in GitHub.

STEPS:

- Login to azure portal, select resource group section.
- Navigate to the deployed KeyVault resource and copy the KeyVault URL from Overview blade.
- This will be the KEYVAULT_NAME secret value. It will be the same for both UI and API
- Navigate to the deployed Azure Container Registry and copy the Login Server name from Overview blade. This will be the ACR_NAME for both UI & API repo secrets.
- Navigate to the deployed Azure Container Registry and copy the Username & Password from Access Keys blade. This will be the ACR_USERNAME & ACR_PASSWORD for both UI & API repo secrets.
- Navigate to the deployed App Service for SDR API and copy the name from Overview blade. This will be the AZURE_WEBAPP_NAME for API repo secrets.
- Navigate to the deployed Function App Service for SDR API and copy the name from Overview blade. This will be the AZURE_FUNCTIONAPP_NAME for API repo secrets.
- Navigate to the deployed App Service for SDR UI and copy the name from Overview blade. This will be the AZURE_WEBAPP_NAME for UI repo secrets.
- The value to be added in AZURE_SP secret is generated during Infra Deployment during App Registration and is available in Key Vault secret with name **Azure-SP**. Retrieve this value to add to GitHub.
- Login to GitHub and navigate to API Repo → Settings → Secrets → Actions and add/replace values for AZURE_SP, ACR_NAME, ACR_USERNAME, ACR_PASSWORD, AZURE_FUNCTIONAPP_NAME, AZURE_WEBAPP_NAME and KEYVAULT_NAME by clicking on update.
- Login to GitHub and navigate to UI Repo → Settings → Secrets → Actions and add/replace values for AZURE_SP, ACR_NAME, ACR_USERNAME, ACR_PASSWORD, AZURE_WEBAPP_NAME and KEYVAULT_NAME by clicking on Update.

4.2 Deploy the UI Application

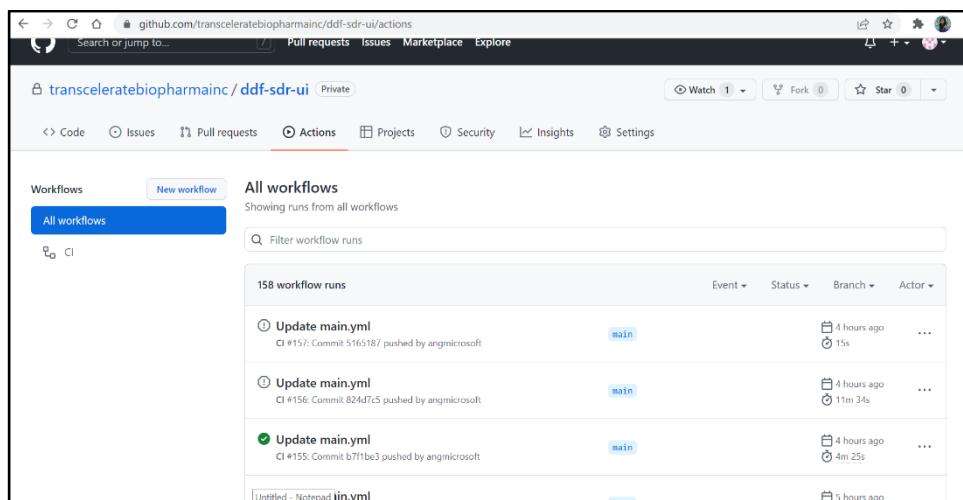
PRE-REQUISITES

- Contributor level of access at Resource Group level.

DEPLOYMENT STEPS:

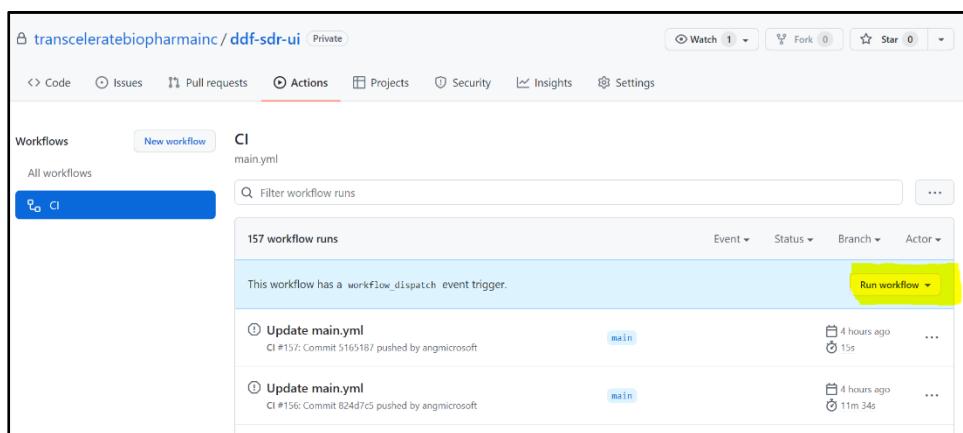
- Go to UI repository on GitHub.
- Click on Actions tab.

Figure 44 SDR UI Repo - GitHub Actions



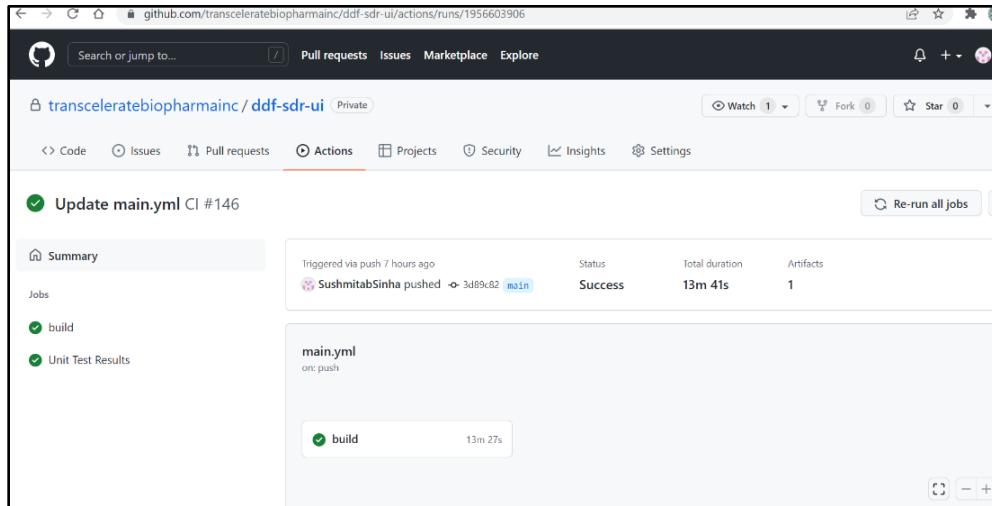
- Click on the workflow CI under All workflow.

Figure 45 GitHub Actions - CI Workflow



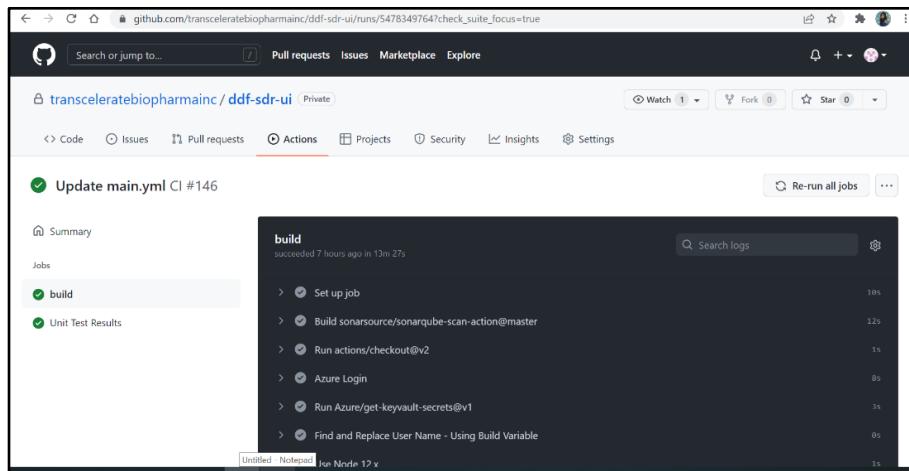
- Click on Run Workflow on the right-hand side. Then the action will be triggered.

Figure 46 GitHub - CI Workflow Run



v. The build logs can be seen on clicking the active/running action.

Figure 47 GitHub CI Workflow Output



vi. On completion of the workflow, the UI application will be deployed to Azure App Service.

4.3 Deploy Back-End API and Function App

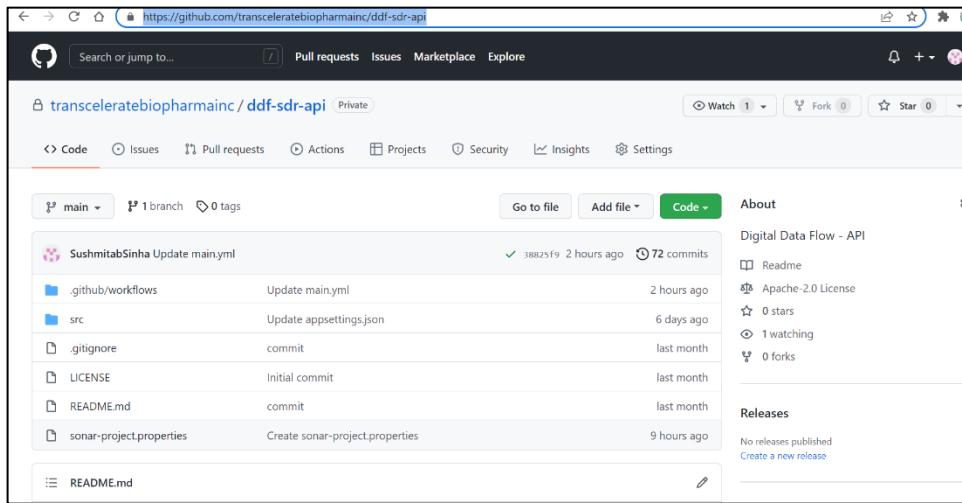
PRE-REQUISITES

- Contributor access at Resource group Level.

DEPLOYMENT STEPS:

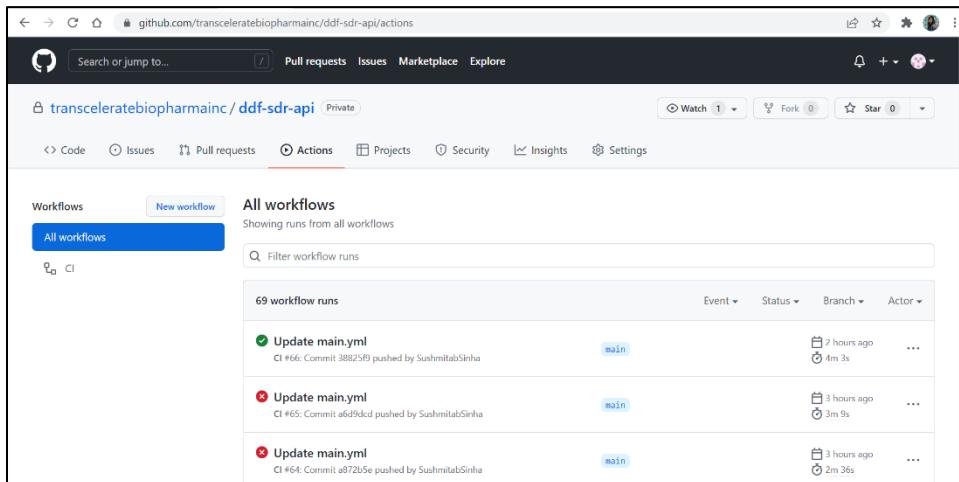
- Go to API repository on GitHub.

Figure 48 GitHub SDR API Repo



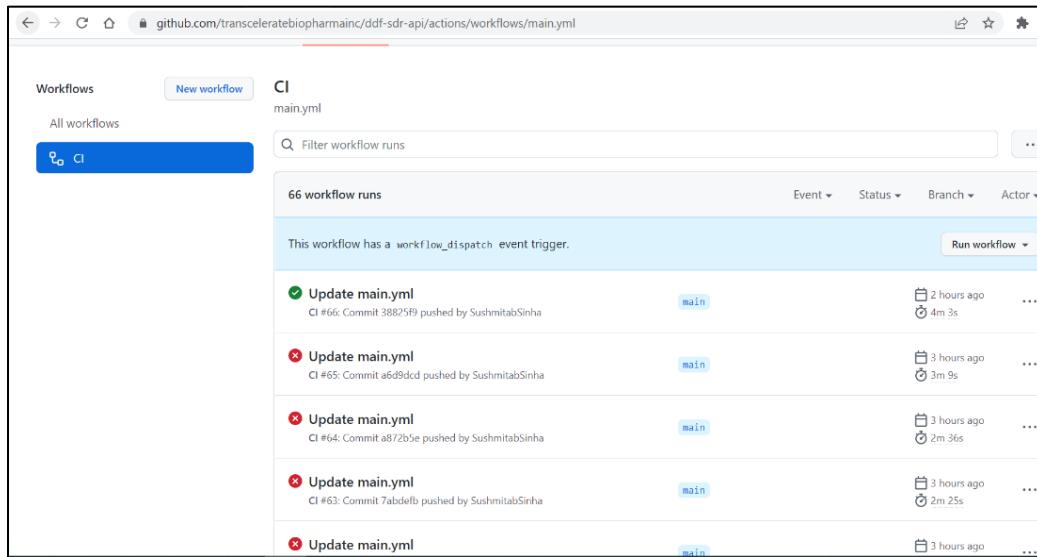
ii. Click on Actions tab.

Figure 49 GitHub Actions CI Workflow



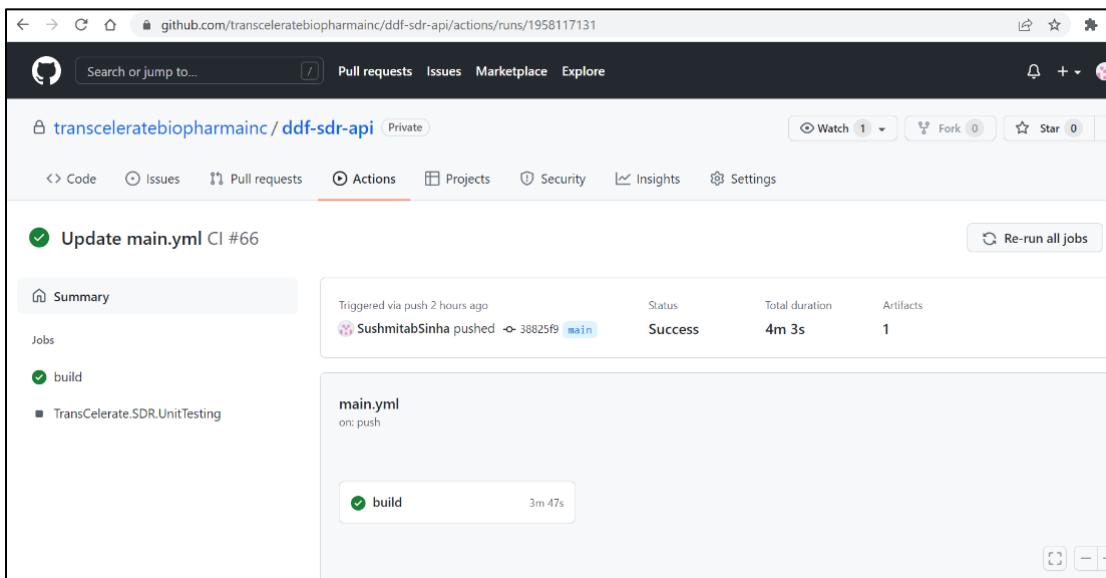
iii. Click on the workflow CI under All workflow.

Figure 50 GitHub CI Workflow Run



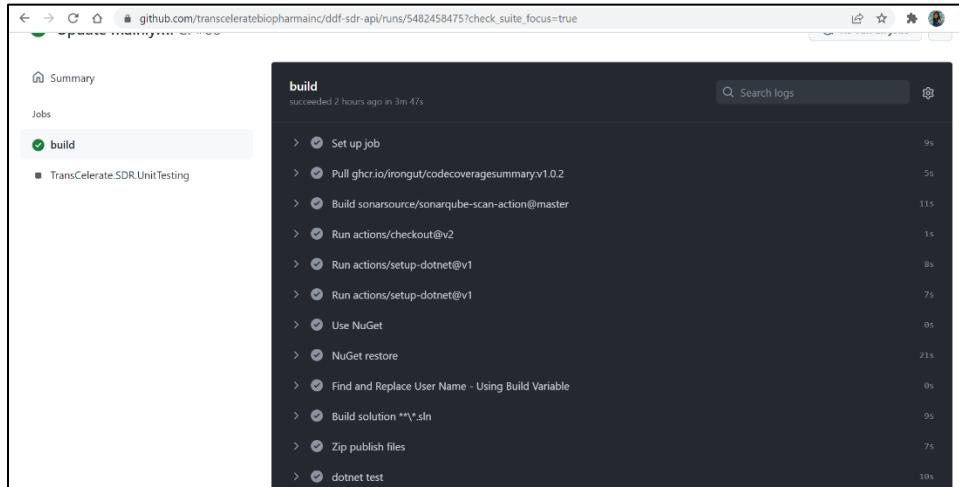
iv. Click on Run Workflow on the right-hand side. Then the action will be triggered.

Figure 51 GitHub CI Workflow Run



v. The build logs can be viewed on clicking the active/running action.

Figure 52 GitHub CI Workflow Output



vi. On completion of the workflow, the back-end API will be deployed to Azure App Service.

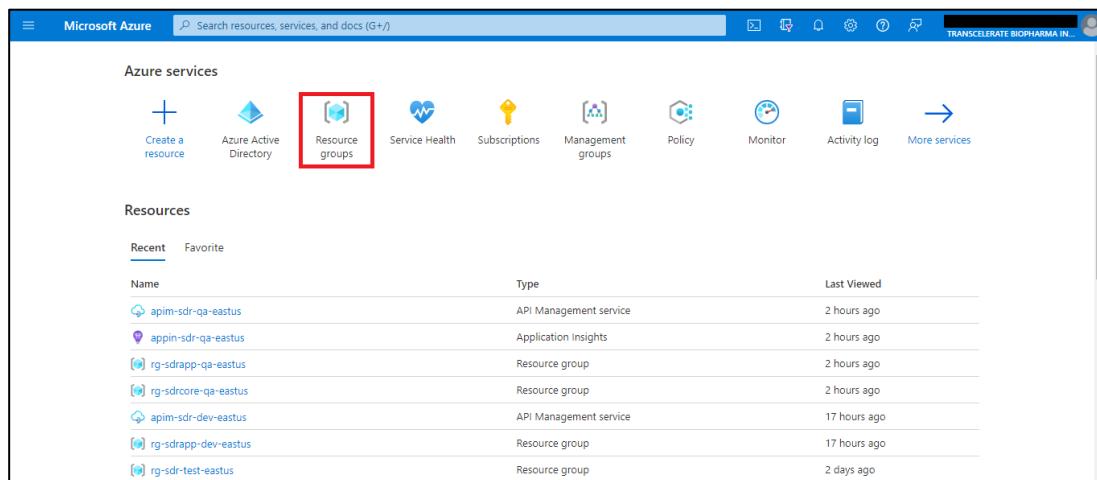
4.4 Deployment Verification

UI APPLICATION VERIFICATION STEPS:

This is to verify the UI Application deployment was successful.

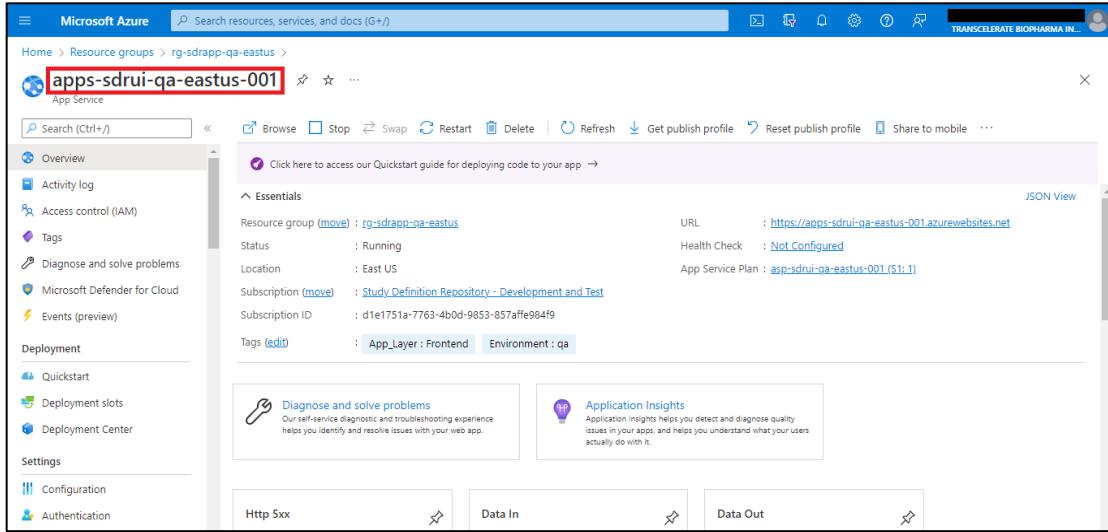
- Go to portal.azure.com. Click on resource group.

Figure 53 Azure Portal



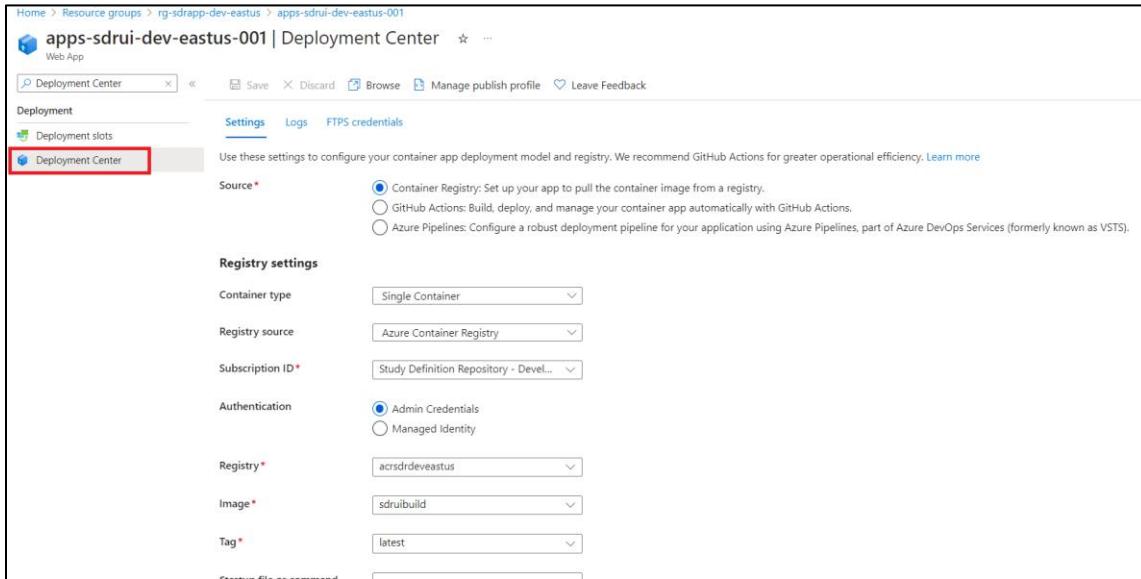
- Select the required resource group and select the UI App Service instance.

Figure 54 Azure Portal - SDR UI App Service



iii. In search box, search for Deployment Center.

Figure 55 SDR UI App Service - Deployment Center Details



iv. Click on Logs.

Figure 56 SDR UI App Service Deployment Center -Logs

SDR API BACK-END APP AND FUNCTION APP VERIFICATION:

The same steps as mentioned above for SDR UI Application verification can be followed for SDR API deployment verification as well, in the corresponding App Service instance.

FUNCTION APP VERIFICATION STEPS:

This is to verify the Function App deployment was successful.

- i. Go to portal.azure.com. Click on resource group.

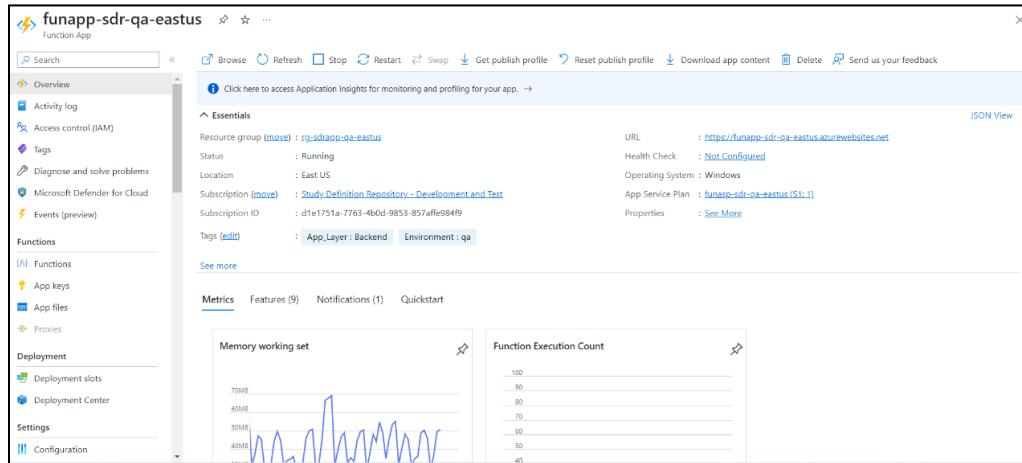
Figure 57 Azure Portal

The screenshot shows the Microsoft Azure portal's main dashboard. At the top, there's a search bar and a navigation bar with icons for Home, Notifications, and Profile. Below the search bar, the title "Microsoft Azure" is displayed. The main content area is titled "Azure services" and contains several icons: "Create a resource" (plus sign), "Azure Active Directory", "Resource groups" (highlighted with a red box), "Service Health", "Subscriptions", "Management groups", "Policy", "Monitor", "Activity log", and "More services". Below this, a section titled "Resources" shows a "Recent" tab with a list of resources. The list includes: "apim-sdr-qa-eastus" (API Management service), "appn-sdr-qa-eastus" (Application insights), "rg-sdrapp-qa-eastus" (Resource group), "rg-sdrcore-qa-eastus" (Resource group), "apim-sdr-dev-eastus" (API Management service), "rg-sdrapp-dev-eastus" (Resource group), and "rg-sdr-test-eastus" (Resource group). Each item in the list has a small icon, a name, a type, and a "Last Viewed" timestamp.

Name	Type	Last Viewed
apim-sdr-qa-eastus	API Management service	2 hours ago
appn-sdr-qa-eastus	Application insights	2 hours ago
rg-sdrapp-qa-eastus	Resource group	2 hours ago
rg-sdrcore-qa-eastus	Resource group	2 hours ago
apim-sdr-dev-eastus	API Management service	17 hours ago
rg-sdrapp-dev-eastus	Resource group	17 hours ago
rg-sdr-test-eastus	Resource group	2 days ago

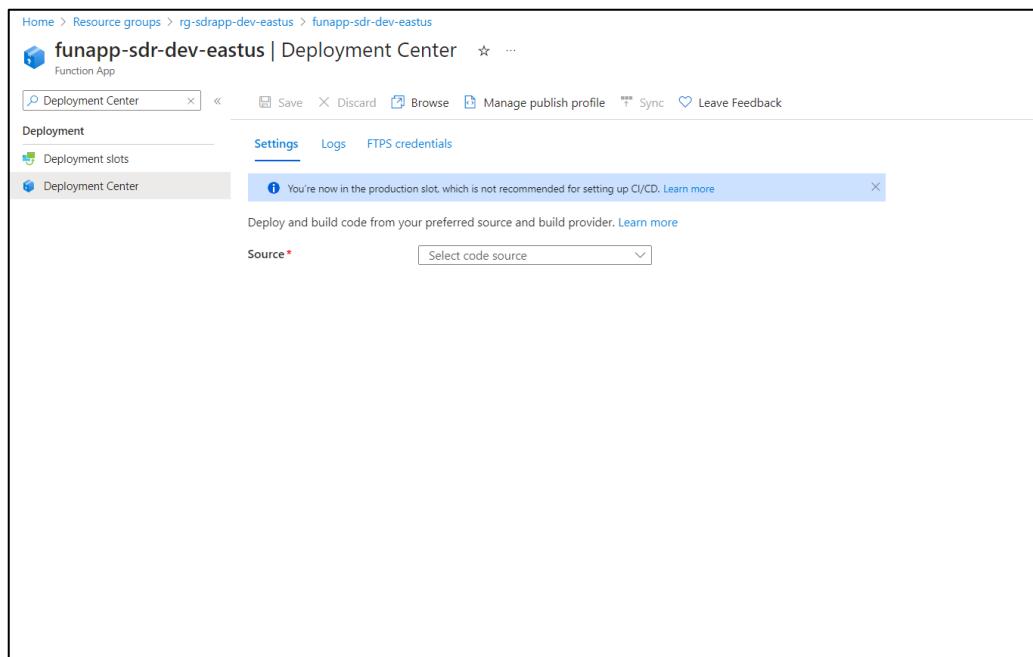
- ii. Select the required resource group and select the Function App Service instance.

Figure 58 Azure Portal - SDR Function App Service



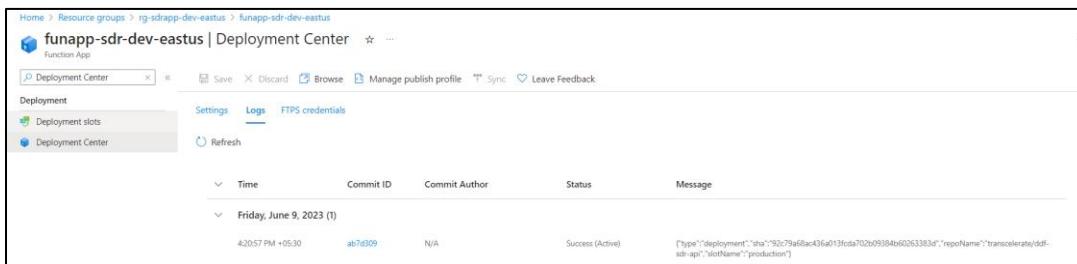
- iii. In search box, search for Deployment Center.

Figure 59 SDR Function App Service – Deployment Center



- iv. Click on Logs

Figure 60 SDR Function App Service Deployment Center – Logs



5. APIM Setup

GOAL:

- API endpoints are grouped into three categories SDR UI API, SDR UI Admin, and SDR API. SDR UI API and SDR UI Admin endpoints will be accessed through SDR UI, and the SDR API endpoints will be accessed through REST clients. The endpoints include CDISC API Spec implementation and other endpoints supporting SDR RI features.
- Configure the Inbound policies on API Endpoints to validate the incoming requests.
- Secure SDR API Endpoint API's using client certificate authentication in API Management
- API Management uses client certificates to secure SDR API Endpoint access (i.e., client to API Management). It will validate certificates presented by the connecting client and compare certificate properties to desired values using policy expressions.

PRE-REQUISITES:

- Contributor access at Resource Group level.

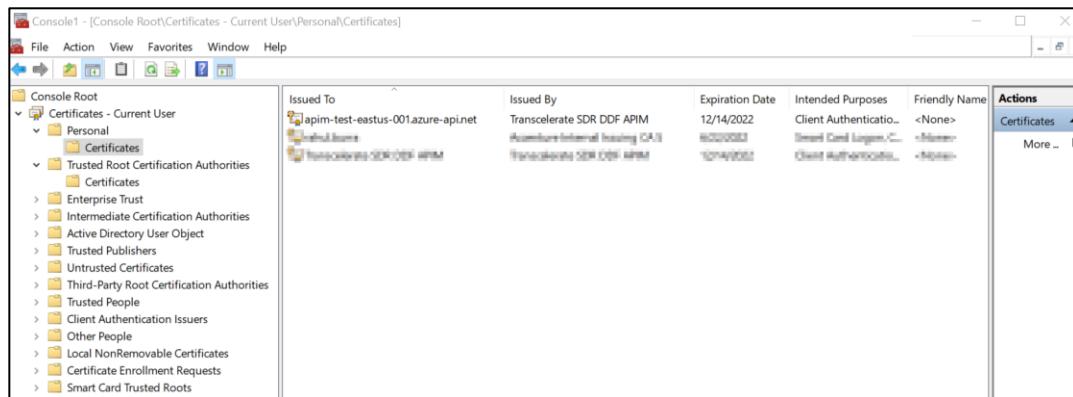
CREATE CLIENT CERTIFICATE:

- i. Create self-signed certificate for authentication.

```
New-SelfSignedCertificate -certstorelocation cert:\CurrentUser\my -dnsname apim-envname-eastus-001.azure-api.net
```

- ii. Once the certificate is created it should now be available to access under your local system snap-in where you can view the metadata of the certificate.

Figure 61 Client Certificate

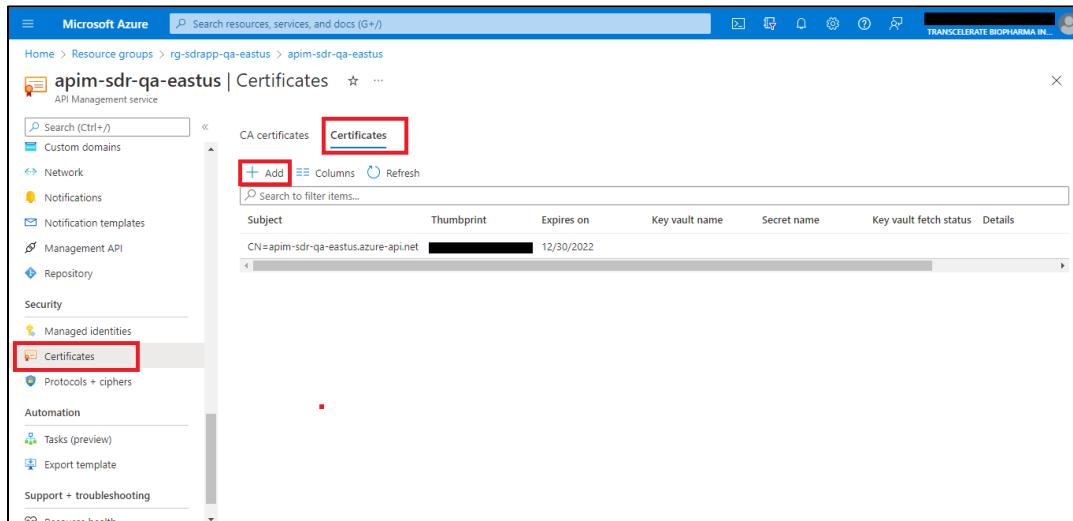


- Export the certificate in .pfx format and set password when prompted.

UPLOAD THE CLIENT CERTIFICATE TO APIM:

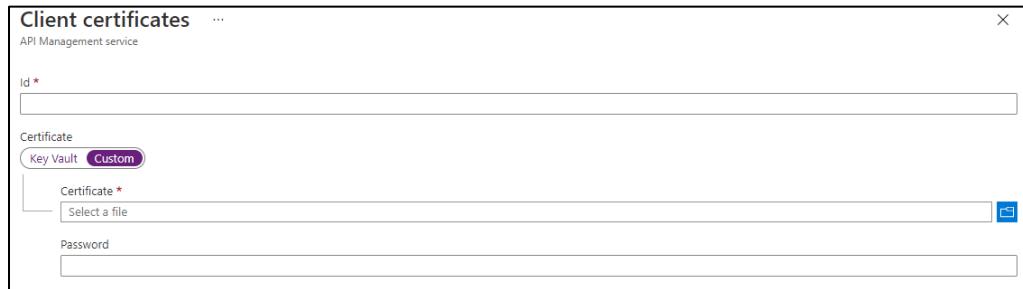
- In Azure Portal, Go to the Certificates option under the “Security” section of APIM. Go to “Certificates” option and click on “Add” option

Figure 62 APIM Certificates



- Upload the password protected Client certificate (.pfx) format as shown below.

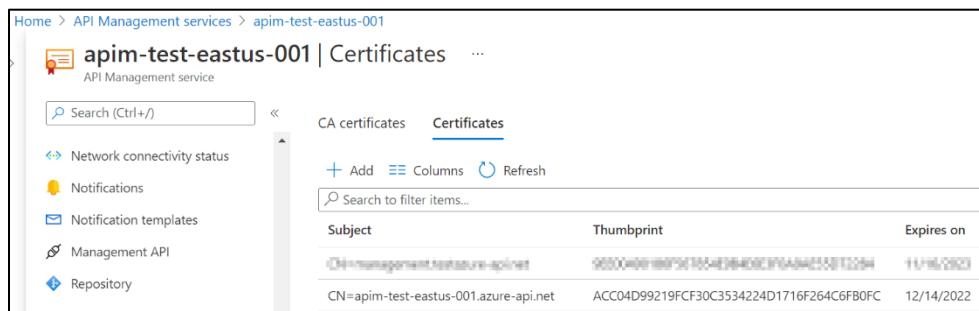
Figure 63 APIM Certificates - Client Certificates



- iii. Once the certificate is uploaded it should be visible on the API Management certificates blade as below

Client Certificate

Figure 64 APIM Certificates - Client Certificate

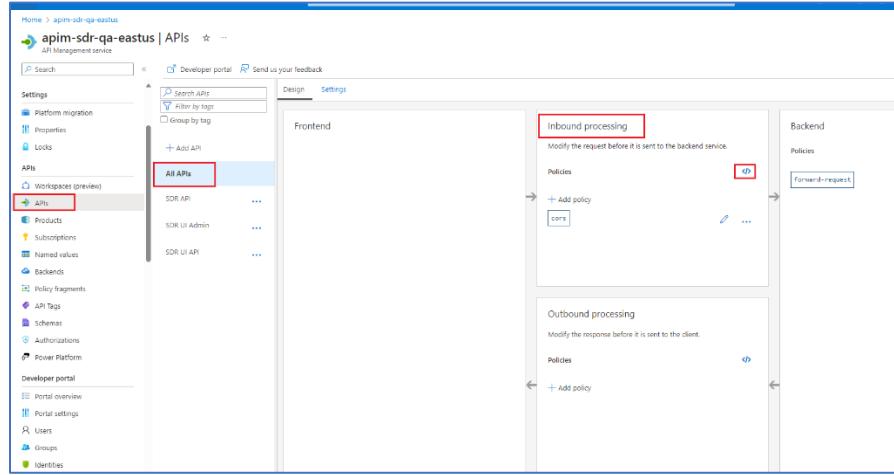


Subject	Thumbprint	Expires on
CN=management.azure-api.net	00000000000000000000000000000000	11/16/2023

CONFIGURE INBOUND POLICY TO ENABLE CORS FOR ALL APIs:

- i. Configure the policy on SDR API to validate one or more attributes of a client certificate used to access APIs hosted in API Management instance.
- ii. Go to APIs -> Select the All APIs -> Select “All Operations” -> Inbound processing -> Select Policies

Figure 65 : APIM Inbound Policy for All APIs



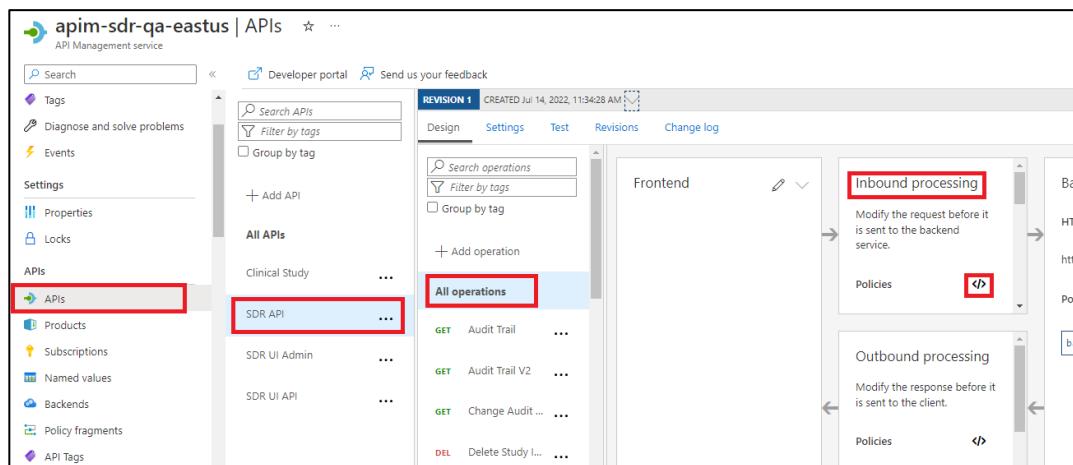
iii. Add the policy code below to validate the Incoming requests.

```
<policies>
<inbound>
    <cors allow-credentials="true">
        <allowed-origins>
            <origin>Add Backend APP Service URL</origin>
            <origin>http://localhost:4200</origin>
            <origin>https://localhost:4200</origin>
            <origin>Add API Management URL</origin>
            <origin>Add Frontend (UI) URL</origin>
            <origin>Add Developer Portal URL</origin>
        </allowed-origins>
        <allowed-methods preflight-result-max-age="300">
            <method>*</method>
        </allowed-methods>
        <allowed-headers>
            <header>*</header>
        </allowed-headers>
        <expose-headers>
            <header>*</header>
        </expose-headers>
    </cors>
</inbound>
<backend>
    <forward-request />
</backend>
<outbound />
<on-error />
</policies>
```

CONFIGURE INBOUND POLICY ON SDR API ENDPOINT FOR CLIENT CERTIFICATE VALIDATION AND API KEY VALIDATION:

- i. Configure the policy on SDR API to validate one or more attributes of a client certificate used to access APIs hosted in API Management instance.
- ii. Go to APIs -> Select the SDR API -> Select “All Operations” -> Inbound processing -> Select Policies

Figure 66 APIM Inbound Processing



- iii. Add the policy code below to check the thumbprint of a client certificate against certificates uploaded to API Management

```
<policies>
<inbound>
    <set-variable name="EmailAddress" value="@{
        string name = "EmptyAuthToken";
        var authHeader =
context.Request.Headers.GetValueOrDefault("Authorization",
"EmptyAuthToken");
        return authHeader.AsJwt()?.Claims.GetValueOrDefault("email",
"EmptyAuthToken");
    }" />
    <set-variable name="UserName" value="@{
        string name = "EmptyAuthToken";
        var authHeader =
context.Request.Headers.GetValueOrDefault("Authorization",
"EmptyAuthToken");
        return authHeader.AsJwt()?.Claims.GetValueOrDefault("name",
"EmptyAuthToken");
    }" />
    <set-variable name="apiKey" value="@{
```

```

        string apiKey = context.Request.Headers.GetValueOrDefault("x-api-key", "EmptyApiKey");
        if((string)apiKey == ""){
            return "EmptyApiKey";
        }
        return apiKey;
    }" />
<set-variable name="subsKeyPrimary" value="@{
    string subKey = context.Subscription?.PrimaryKey;
    return subKey;
}" />
<set-variable name="subsKeySecondary" value="@{
    string subKey = context.Subscription?.SecondaryKey;
    return subKey;
}" />
<choose>
    <when condition="@((context.Variables["EmailAddress"]) != null)">
        <trace source="My Global APIM Policy"
severity="information">
            <message>@(String.Format("{0} | {1}",
context.Api.Name, context.Operation.Name))</message>
            <metadata name="EmailAddress"
value="@((string)context.Variables["EmailAddress"])" />
            <metadata name="UserName"
value="@((string)context.Variables["UserName"])" />
        </trace>
    </when>
    <otherwise>
        <trace source="My Global APIM Policy"
severity="information">
            <message>@(String.Format("{0} | {1}",
context.Api.Name, context.Operation.Name))</message>
            <metadata name="EmailAddress" value="Not Available" />
            <metadata name="UserName" value="Not Available" />
        </trace>
    </otherwise>
</choose>
<base />
<choose>
    <when condition="@((context.Request.Certificate == null ||
!context.Deployment.Certificates.Any(c => c.Value.Thumbprint ==
context.Request.Certificate.Thumbprint)
|| context.Request.Certificate.NotAfter<DateTime.Now))">
        <return-response>
            <set-status code="403" reason="Invalid client
certificate" />
        </return-response>
    </when>
</choose>

```

```

        </when>
    </choose>
    <choose>
        <when condition="@((string)context.Variables["apiKey"] != "EmptyApiKey" && ((string)context.Variables["apiKey"] != (string)context.Variables["subsKeyPrimary"] && (string)context.Variables["apiKey"] != (string)context.Variables["subsKeySecondary"]))">
            <return-response>
                <set-status code="401" reason="Invalid Api Key" />
                <set-header name="content-type" exists-
action="override">
                    <value>application/json</value>
                </set-header>
                <set-body
template="liquid">{"statusCode": "401", "message": "Invalid Api-Key"}</set-
body>
            </return-response>
        </when>
    </choose>

    <cors allow-credentials="true">
        <allowed-origins>
            <origin>Add Backend APP Service URL</origin>
            <origin>http://localhost:4200</origin>
            <origin>https://localhost:4200</origin>
            <origin>Add API Management URL</origin>
            <origin>Add Frontend (UI) URL</origin>
            <origin>Add Developer Portal URL</origin>
        </allowed-origins>
        <allowed-methods preflight-result-max-age="300">
            <method>GET</method>
            <method>POST</method>
            <method>PATCH</method>
            <method>DELETE</method>
        </allowed-methods>
        <allowed-headers>
            <header>*</header>
        </allowed-headers>
        <expose-headers>
            <header>*</header>
        </expose-headers>
    </cors>
</inbound>
<backend>
    <base />
</backend>
<outbound>

```

```

<base />
</outbound>
<on-error>
    <base />
</on-error>
</policies>

```

- iv. Update the “Header name” value to “x-api-key” in APIs -> SDR API -> Settings -> Subscription and click save.

Figure 67 : Update the Api-Key header name

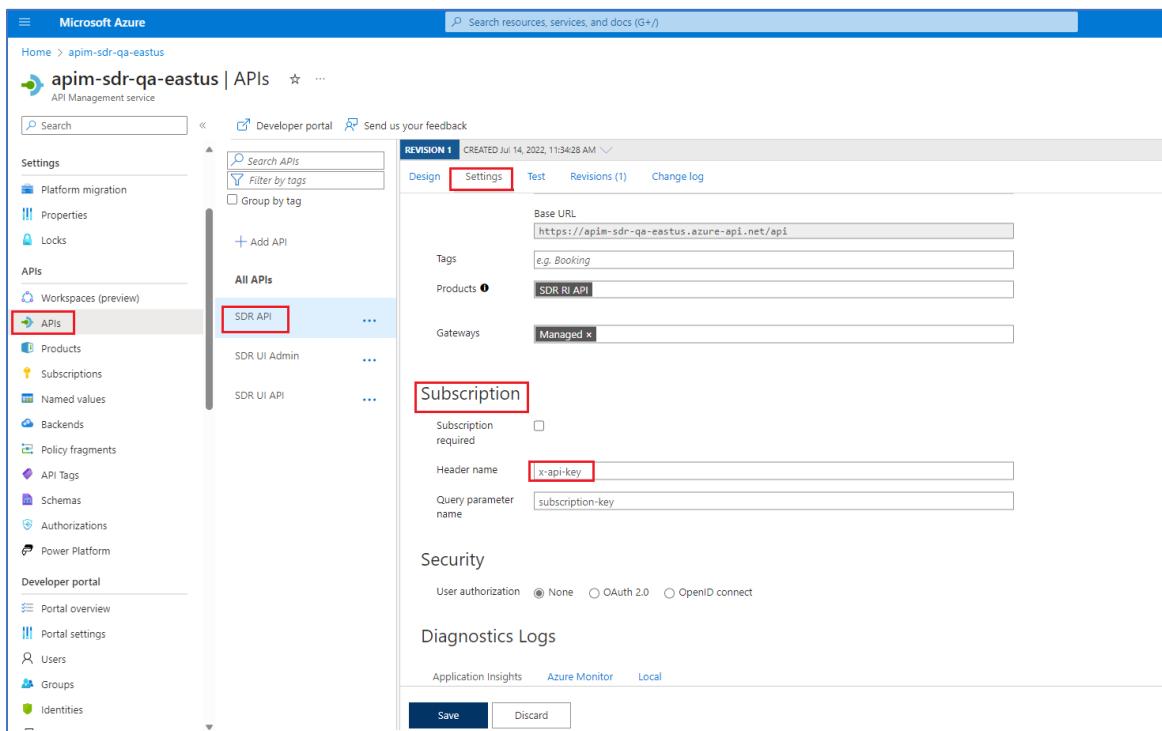
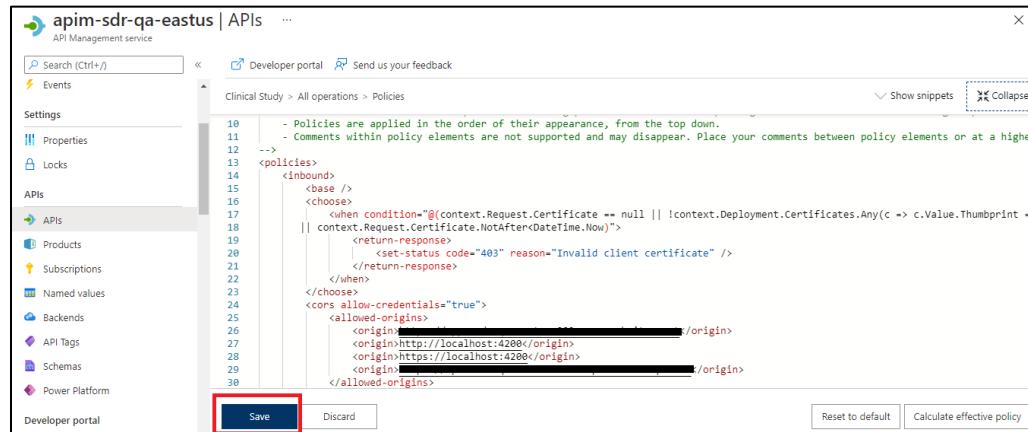
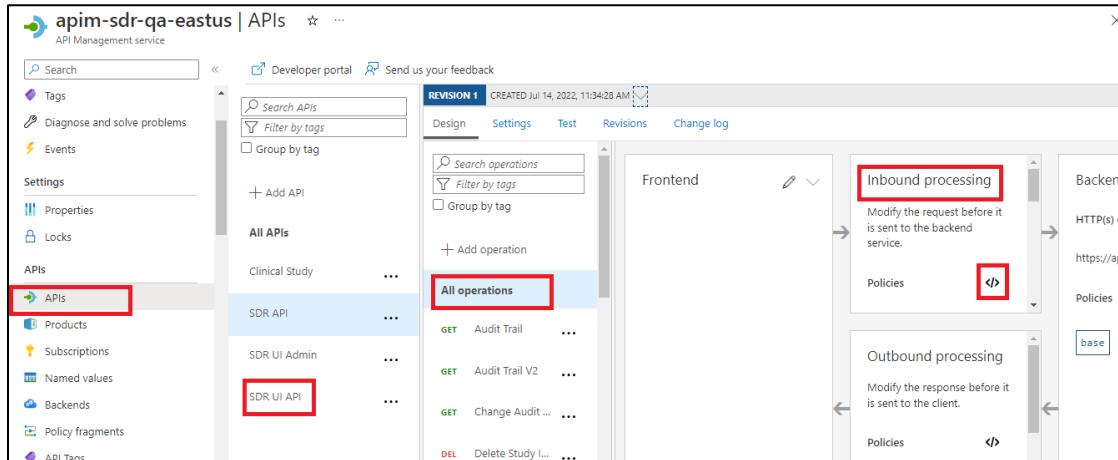


Figure 68 APIM - Inbound Policy



CONFIGURE INBOUND POLICY ON SDR UI API & SDR UI ADMIN ENDPOINTS FOR INCOMING REQUESTS VALIDATION:

- Configure the policy on SDR UI API and SDR UI Admin to validate the inbound requests to access APIs hosted in API Management instance.
- Go to APIs -> Select the SDR UI API -> Select “All Operations” -> Inbound processing -> Select Policies



- Add the policy code below to validate the incoming requests.

```

<policies>
    <inbound>
        <set-variable name="EmailAddress" value="@{
            string name = "EmptyAuthToken";
        }" />

```

```

        var authHeader =
context.Request.Headers.GetValueOrDefault("Authorization",
"EmptyAuthToken");
        return authHeader.AsJwt()?.Claims.GetValueOrDefault("email",
"EmptyAuthToken");
    }" />
<set-variable name="UserName" value="@{
    string name = "EmptyAuthToken";
    var authHeader =
context.Request.Headers.GetValueOrDefault("Authorization",
"EmptyAuthToken");
    return authHeader.AsJwt()?.Claims.GetValueOrDefault("name",
"EmptyAuthToken");
}" />
<choose>
    <when condition="@((context.Variables["EmailAddress"]) != null)">
        <trace source="My Global APIM Policy"
severity="information">
            <message>@(String.Format("{0} | {1}",
context.Api.Name, context.Operation.Name))</message>
            <metadata name="EmailAddress"
value="@((string)context.Variables["EmailAddress"])" />
            <metadata name="UserName"
value="@((string)context.Variables["UserName"])" />
        </trace>
    </when>
    <otherwise>
        <trace source="My Global APIM Policy"
severity="information">
            <message>@(String.Format("{0} | {1}",
context.Api.Name, context.Operation.Name))</message>
            <metadata name="EmailAddress" value="Not Available" />
            <metadata name="UserName" value="Not Available" />
        </trace>
    </otherwise>
</choose>
<base />
<cors allow-credentials="true">
    <allowed-origins>
        <origin>Add Backend APP Service URL</origin>
        <origin>http://localhost:4200</origin>
        <origin>https://localhost:4200</origin>
        <origin>Add API Management URL</origin>
        <origin>Add Frontend (UI) URL</origin>
    </allowed-origins>
    <allowed-methods preflight-result-max-age="300">
        <method>GET</method>
    </allowed-methods>
</cors>

```

```

<method>POST</method>
<method>PATCH</method>
<method>DELETE</method>
</allowed-methods>
<allowed-headers>
    <header>*</header>
</allowed-headers>
<expose-headers>
    <header>*</header>
</expose-headers>
</cors>
</inbound>
<backend>
    <base />
</backend>
<outbound>
    <base />
</outbound>
<on-error>
    <base />
</on-error>
</policies>

```

- iv. Repeat the above steps to configure the inbound policy on SDR UI Admin endpoint.



5.1. Developer Portal Configuration

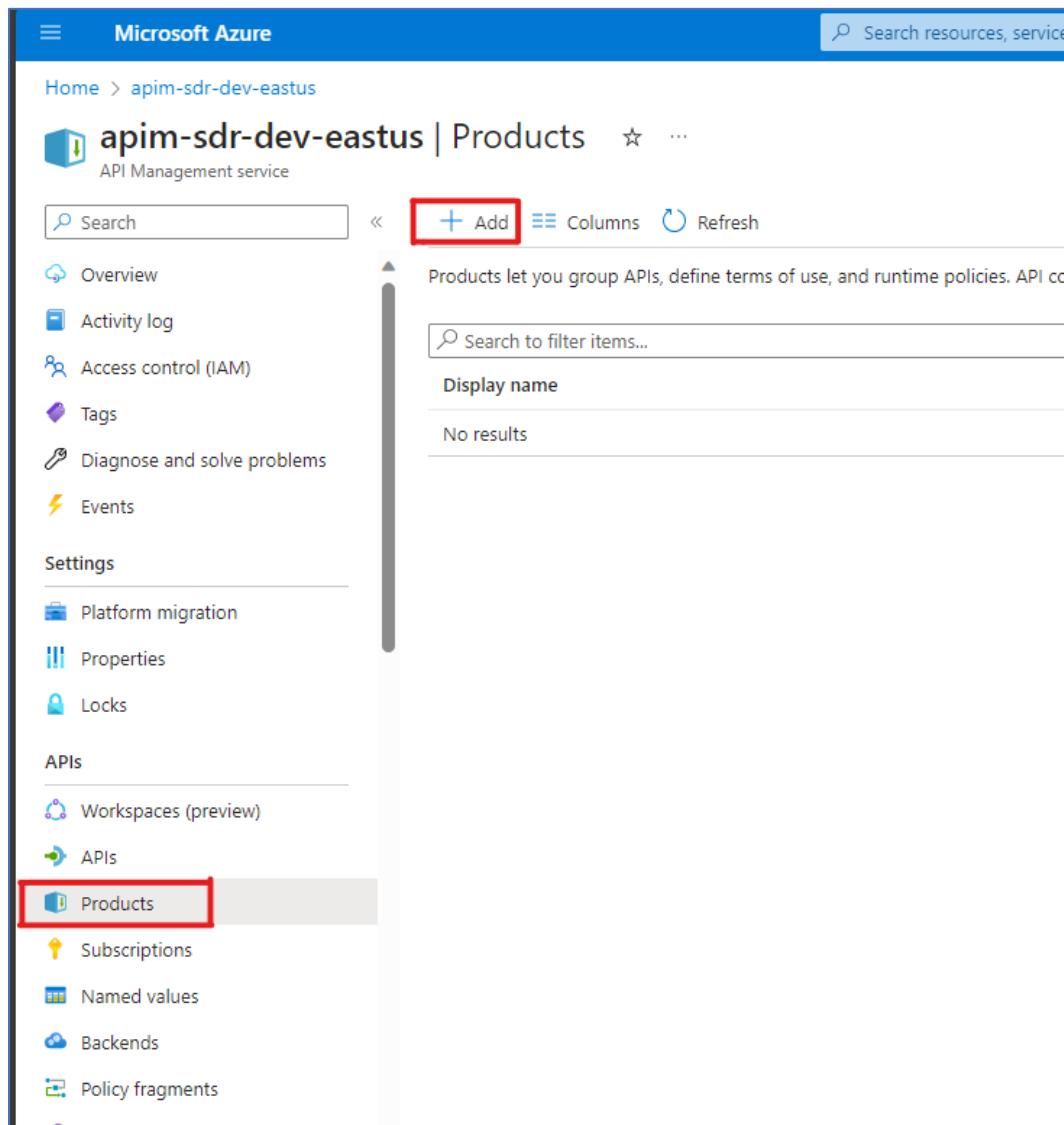
APIM Developer Portal can be used to generate API-Key for SDR API access. For accessing the developer portal and generating the API-Key, a few steps need to be followed.

An API product needs to be created and an Azure AD group access must be configured in the product. Then, the users under the group can access the developer portal and generate an API-Key. Below are the steps to be followed.

5.1.1. Creating a Product

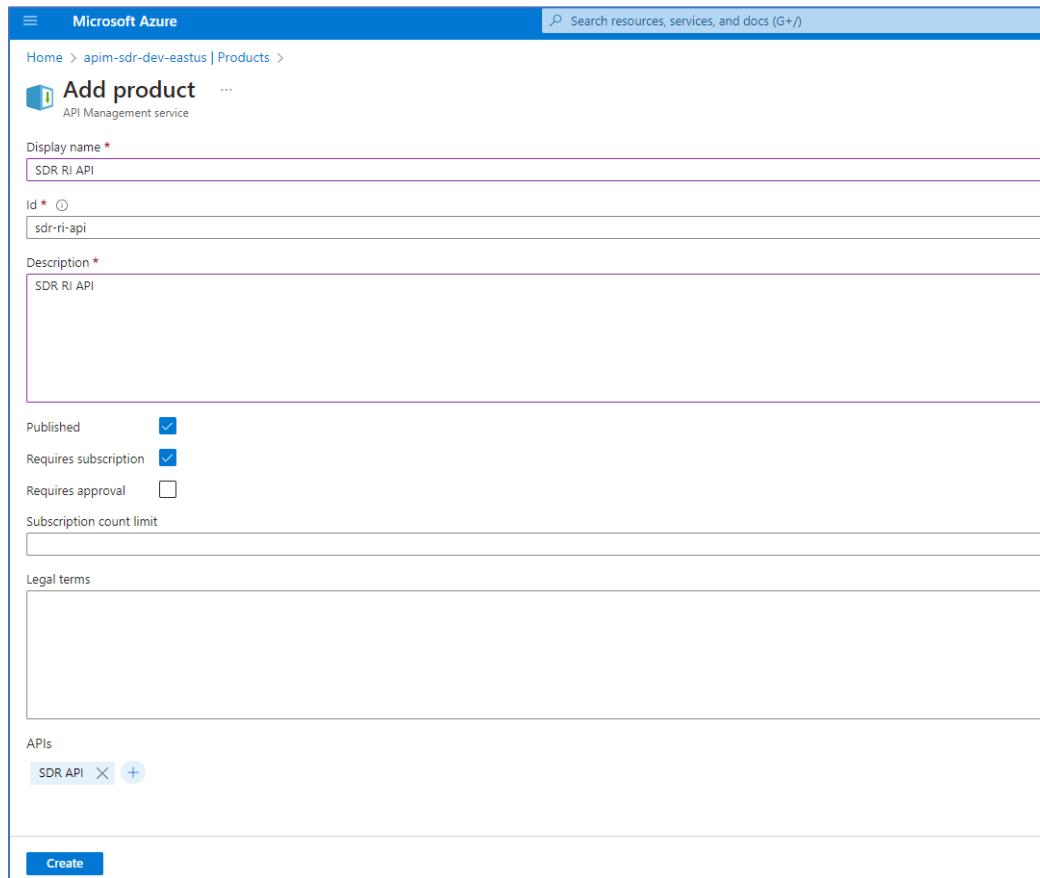
- i. In the Azure Portal, goto APIM and select Products.
- ii. Select Add to create a Product.

Figure 69 : Add Product



- iii. On the Add screen, Add Display Name and Description.
- iv. Check **Published** and **Requires subscription** options.
- v. Under APIs click Add and select SDR API
- vi. Then click Create.

Figure 70 : Add Product



The screenshot shows the 'Add product' page in the Microsoft Azure API Management service. The form fields are as follows:

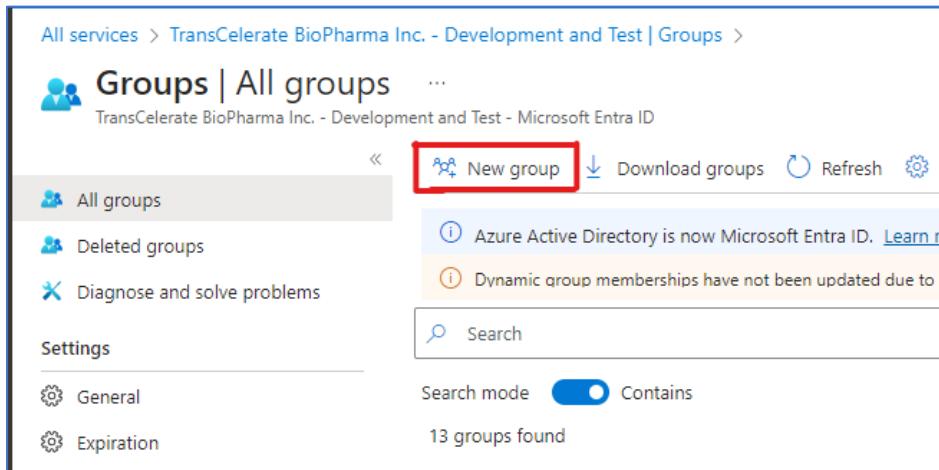
- Display name ***: SDR RI API
- Id ***: sdr-ri-api
- Description ***: SDR RI API
- Published**:
- Requires subscription**:
- Requires approval**:
- Subscription count limit**: (empty)
- Legal terms**: (empty)
- APIs**: SDR API (selected) with a remove (X) and add (+) button.

A blue 'Create' button is at the bottom right.

5.1.2. Azure AD Group Creation

- i. In the Azure Portal, select Microsoft Entra ID.
- ii. Select Groups under the left menu.
- iii. Select New Group option in the All Groups screen.

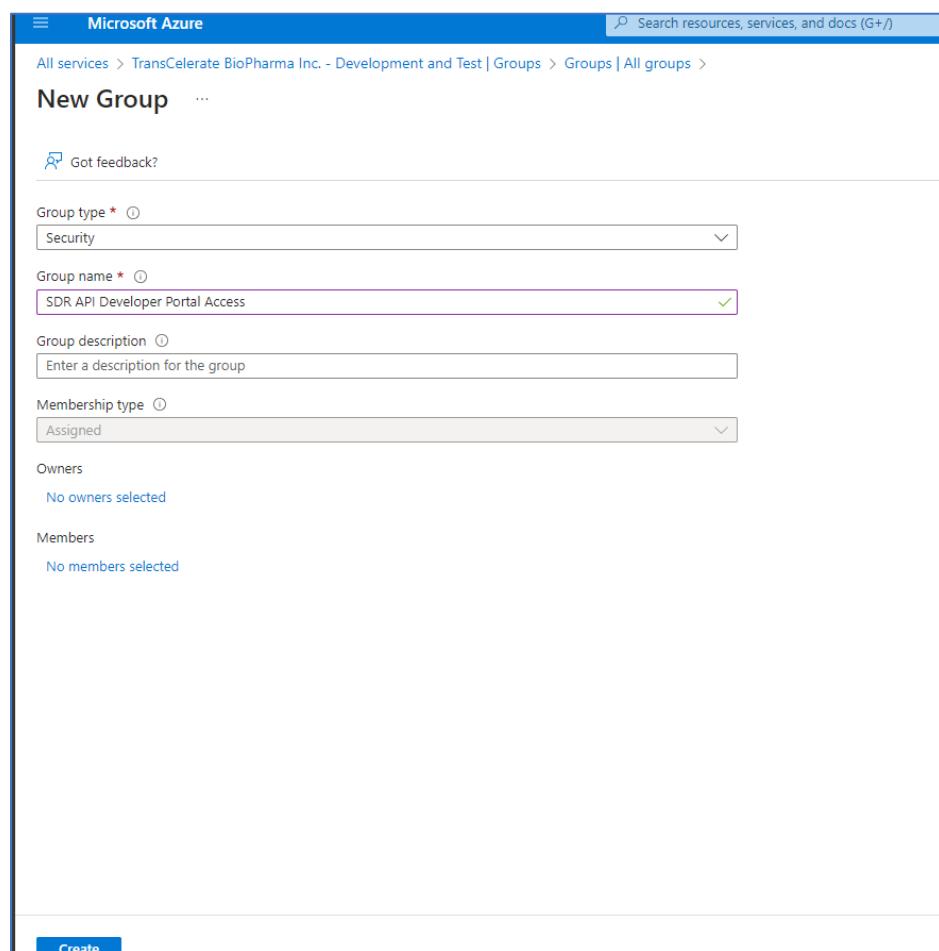
Figure 71 : Azure AD Groups



The screenshot shows the 'Groups | All groups' page in the Azure portal. The top navigation bar includes 'All services > TransCelerate BioPharma Inc. - Development and Test | Groups'. Below the title, there's a message about Microsoft Entra ID. On the left, a sidebar lists 'All groups', 'Deleted groups', 'Diagnose and solve problems', 'Settings', 'General', and 'Expiration'. The main area shows a search bar and a message indicating dynamic group memberships are not updated. A red box highlights the 'New group' button in the top right corner.

iv. In the New Group screen, add group name and click create.

Figure 72 : Add New Azure AD Group



The screenshot shows the 'New Group' creation form in the Azure portal. The top navigation bar includes 'All services > TransCelerate BioPharma Inc. - Development and Test | Groups > Groups | All groups'. The form fields are as follows:

- Group type ***: Security (selected)
- Group name ***: SDR API Developer Portal Access (highlighted with a purple border)
- Group description**: Enter a description for the group (empty)
- Membership type**: Assigned (selected)
- Owners**: No owners selected
- Members**: No members selected

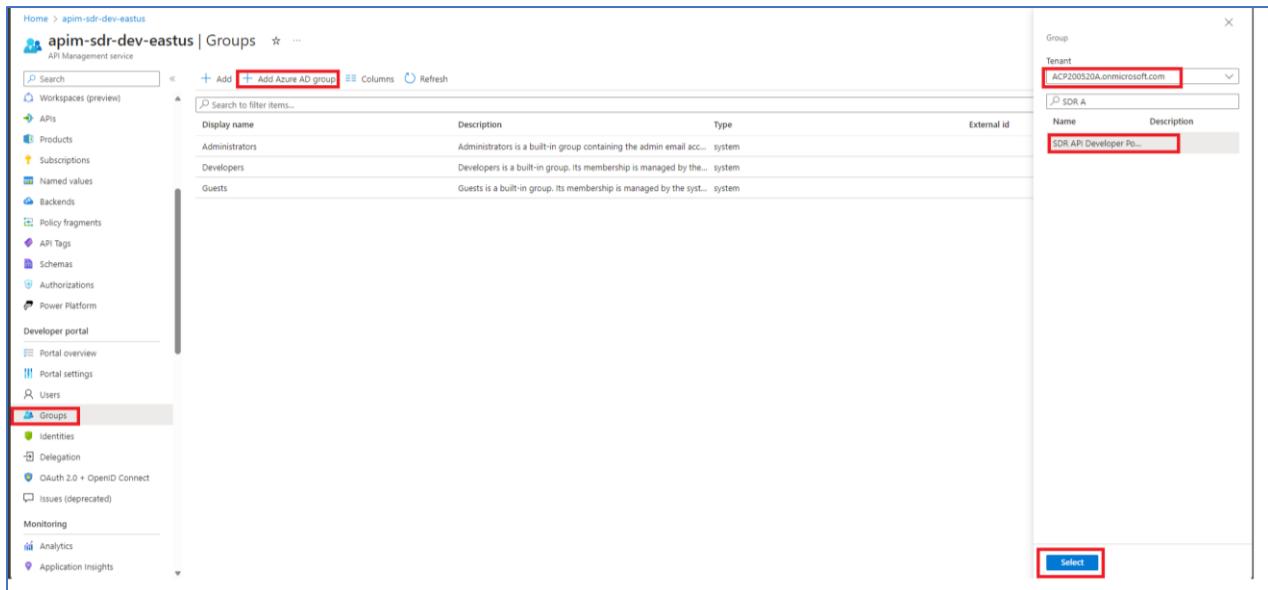
A blue 'Create' button is at the bottom of the form.

- v. In the New Group screen, add group name and click create.

5.1.3. Add Azure AD Group in APIM

- i. In Azure Portal, goto APIM and select Groups.
- ii. Select Add Azure AD group option.
- iii. On click, select the Tenant.
- iv. Search and select the group created for Developer portal access.
- v. Then click Select.
- vi. This will add Azure AD Group in APIM.

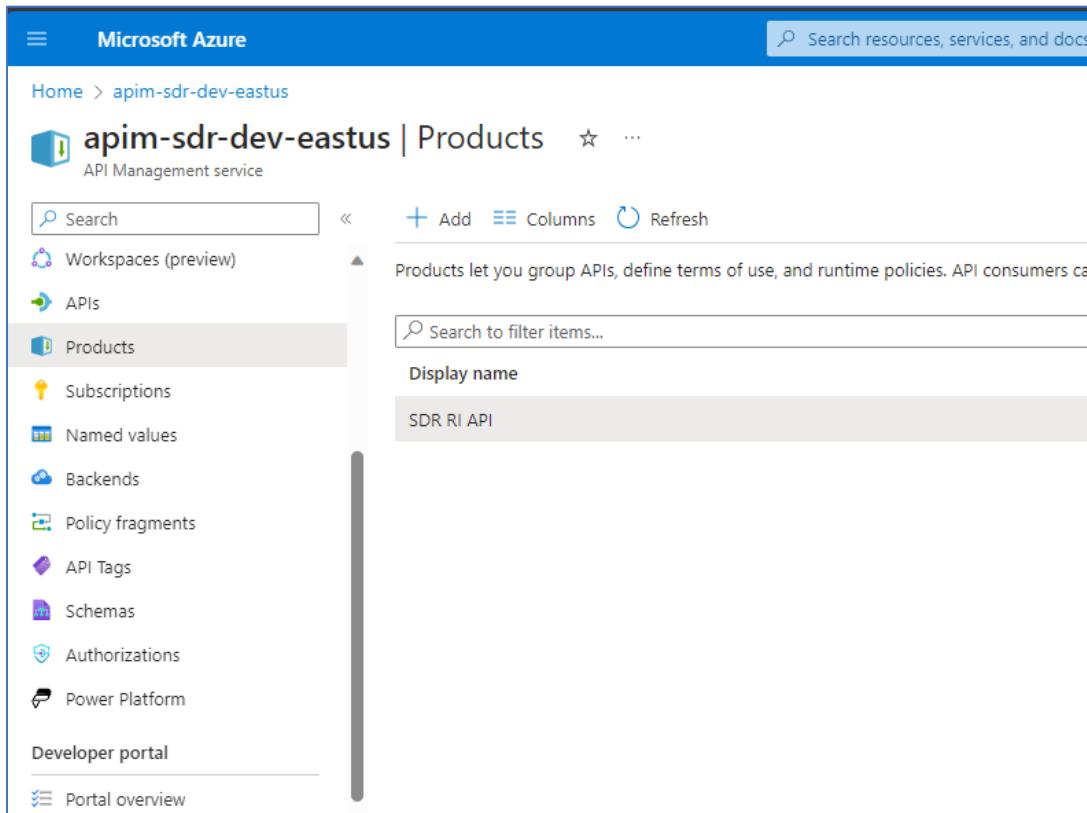
Figure 73 : Add Azure AD Group in APIM



5.1.4. Add Azure AD Group in Product

- i. In Azure Portal, goto APIM and select Products.
- ii. Then select the SDR RI API Product.

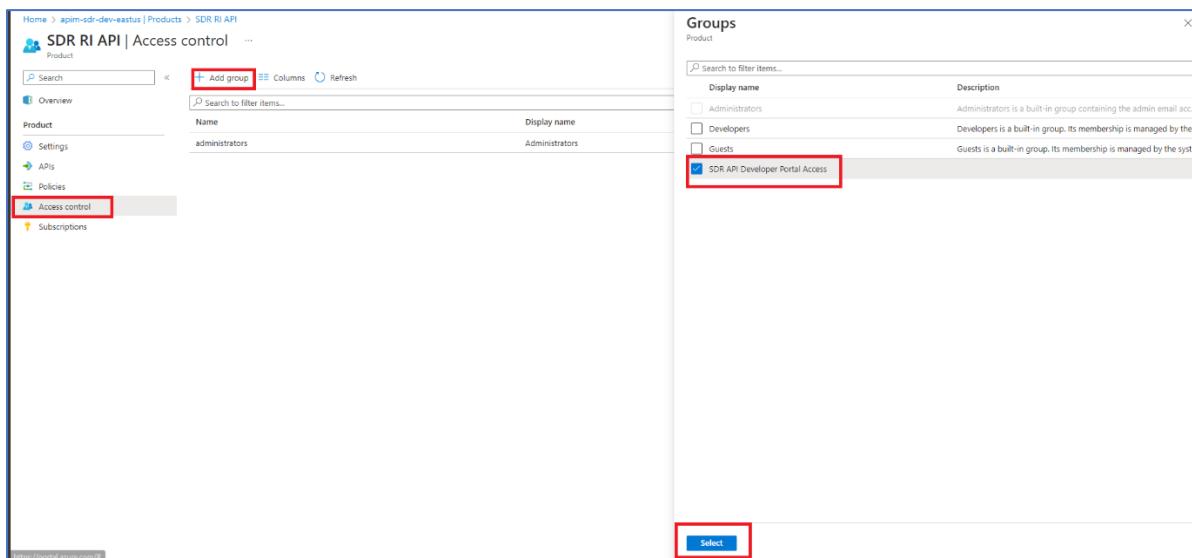
Figure 74 : SDR RI API Product in APIM



The screenshot shows the Microsoft Azure API Management interface. The top navigation bar includes the TransCelerate logo, the page title "SDR RI Platform Setup and Deployment Guide", and a search bar. The main content area is titled "apim-sdr-dev-eastus | Products". On the left, a sidebar lists various management options: Workspaces (preview), APIs, Products (selected), Subscriptions, Named values, Backends, Policy fragments, API Tags, Schemas, Authorizations, Power Platform, Developer portal, and Portal overview. The main pane displays a product named "SDR RI API" with a display name of "SDR RI API". A note states: "Products let you group APIs, define terms of use, and runtime policies. API consumers can access your APIs through the developer portal or directly via the API endpoint." Below the product listing is a search bar.

- iii. Then under Access Control, select Add group.
- iv. Then select the AD Group added and click select.

Figure 75 : Add group access in SDR RI API Product



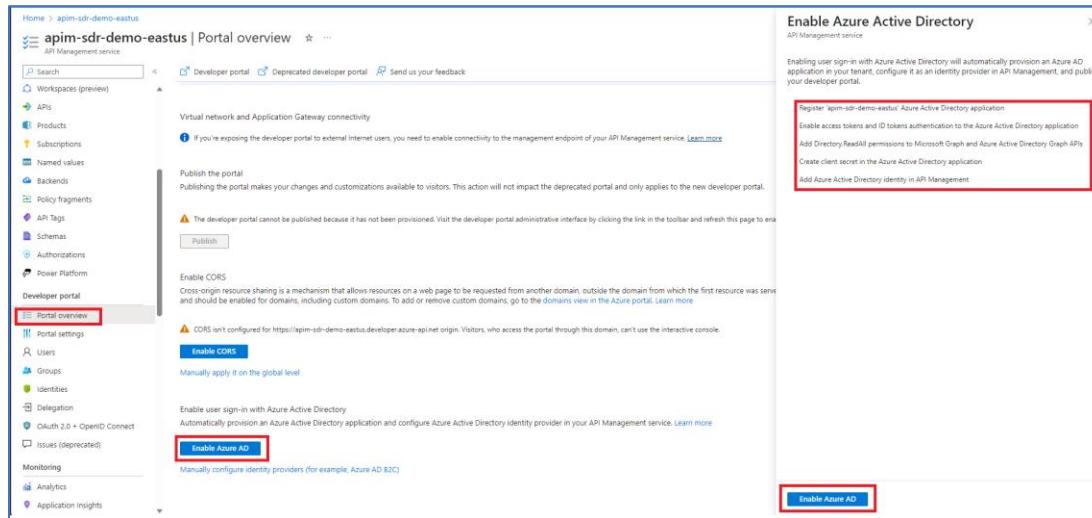
The screenshot shows the "Access control" page for the "SDR RI API" product. The left sidebar shows "Overview", "Product", "Settings", "APIs", "Policies", and "Access control" (which is selected). The main pane has a search bar and an "Add group" button. A list of groups is shown, with one item selected: "SDR API Developer Portal Access". At the bottom right, there is a "Select" button.

Name	Display name
administrators	Administrators
<input checked="" type="checkbox"/> SDR API Developer Portal Access	

5.1.5. Add App registration for Azure AD login

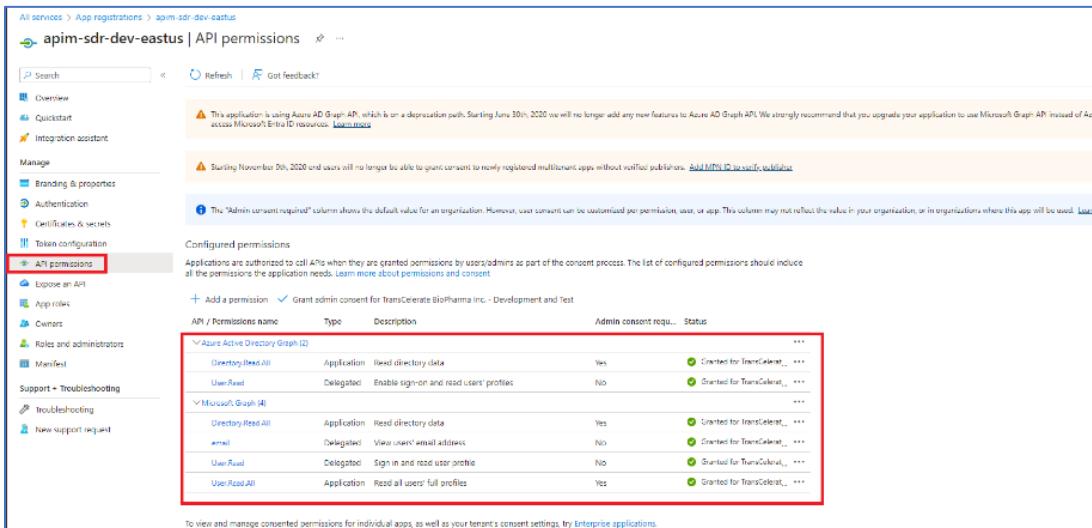
- i. In Azure Portal, goto APIM and select Portal Overview.
- ii. Select Enable Azure AD button.
- iii. On click, the Enable Azure Active Directory dialog will show the list of activites that will be done as a part of enabling the Azure AD.
- iv. Click Enable Azure AD button.

Figure 76 : Enable Azure AD for Developer Portal



- v. After enabling the Azure AD, a new app registration will be created.
- vi. In the Azure Portal, search and select App Registration.
- vii. Under All applications, select the application with the same name as APIM.
- viii. Select API Permissions option to add required permissions.
- ix. Add email with Delegated permission and User.Read.All with Application permission under Microsoft.Graph section.

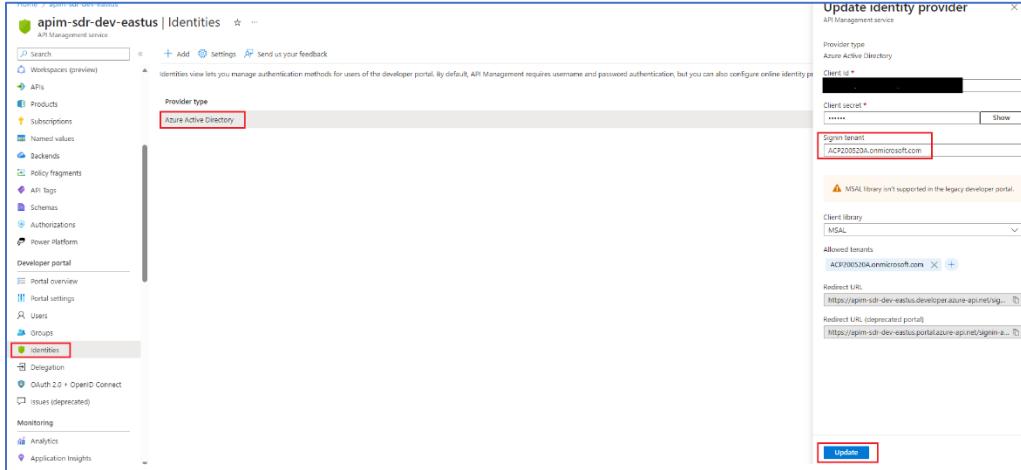
Figure 77 : API Permissions for APIM App Registration



5.1.6. Identities in APIM

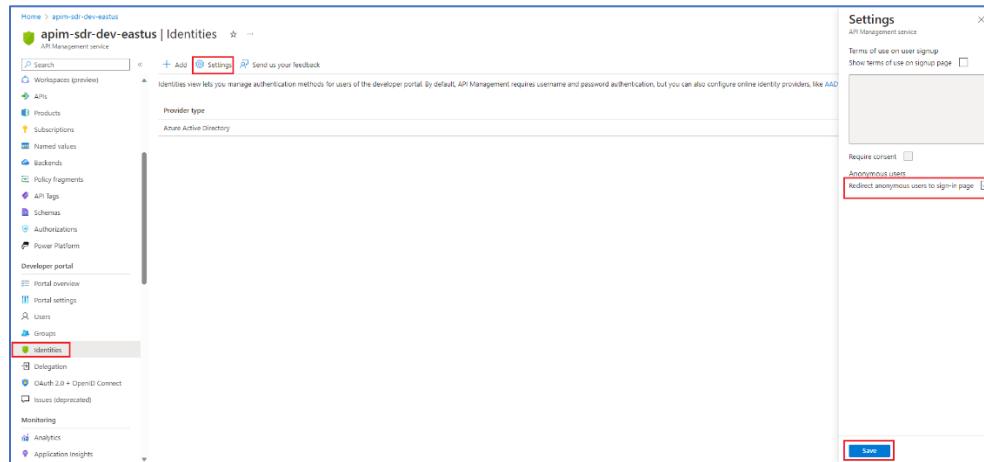
- i. In Azure Portal, goto APIM and select Identities.
- ii. Select Azure Active Directory.
- iii. Add the primary domain in the Signin tenant value.
- iv. Then click Update.

Figure 78 : Add Sign-in tenant



- v. Under same Identities section, select Settings.
- vi. Check Redirect anonymous users to sign-in page and click save.
- vii. This option will restrict unauthorized users to redirect to Signin page.

Figure 79 : Redirect anonymous users to sign-in page

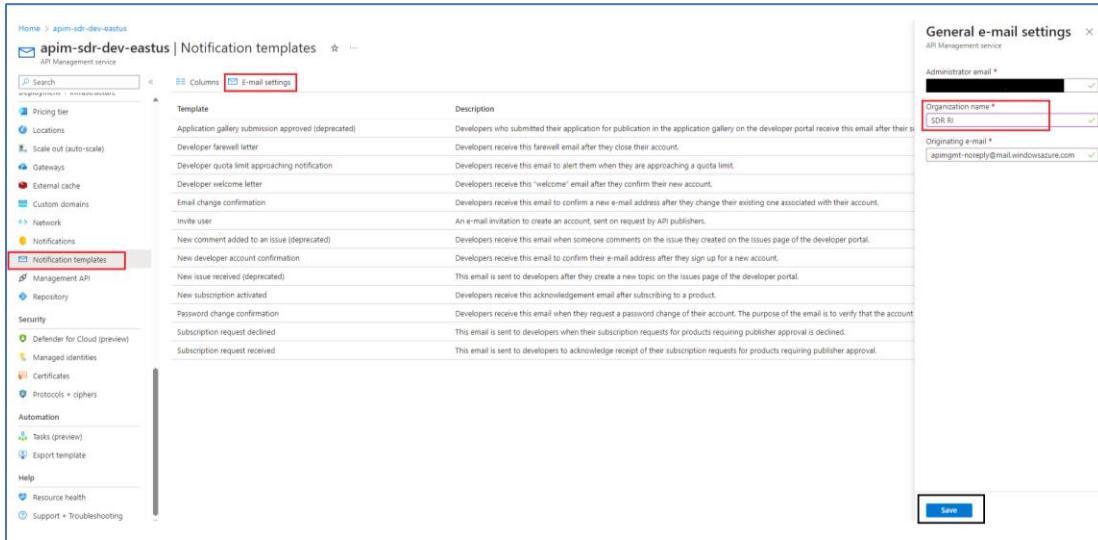


5.1.7. Change Publisher Name in APIM

- i. In Azure Portal, goto APIM and select Notification Templates.

- ii. Select E-mail Settings and update the Organization name to update the publisher name of the APIs.
- iii. Then click save.

Figure 80 : Change Publisher Name

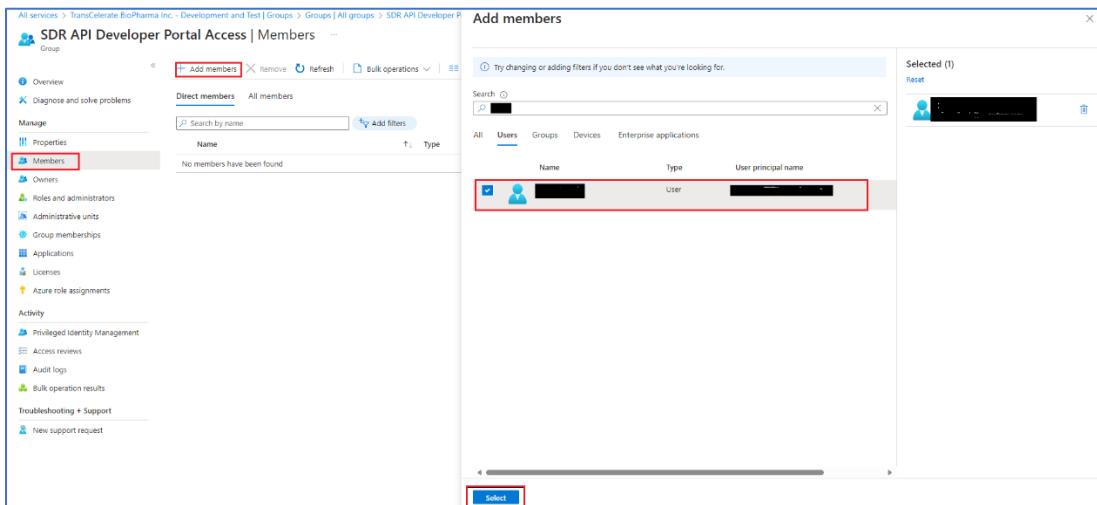


The screenshot shows two windows side-by-side. On the left is a list of notification templates with one item selected: 'Application gallery submission approved (deprecated)'. On the right is a 'General e-mail settings' dialog box with fields for 'Administrator email', 'Organization name' (set to 'SDR RI'), and 'Originating e-mail'. A 'Save' button is at the bottom right of the dialog.

5.1.8. Add User to the AD Group for Developer Portal Access

- i. In Azure Portal, goto Microsoft Entra ID and click groups.
- ii. Select the group created for developer portal access.
- iii. Then click members.
- iv. Then click Add members to add users to the Azure AD Group.
- v. Then under Users, search for the user and select the user.
- vi. Then click select to add the user to the Azure AD Group.

Figure 81 : Add User to Azure AD Group



The screenshot shows the 'Add members' dialog box in the Azure portal. It lists a single user named '██████████' under the 'Users' tab. A red box highlights the 'Select' button at the bottom of the list.

Note: The developer portal must be customized to meet the specific needs.