



# **Study Definition Repository (SDR) Reference Implementation**

## **Setup and Deployment Guide Version 7.0**

### **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.
V7.0	22-Apr-2023	ACN	<ul style="list-style-type: none"> <li>• Added steps to create default group in Groups collection of Cosmos DB.</li> <li>• APIM stv2 platform related changes updated.</li> <li>• Updated developer portal configuration section to align with the latest terraform scripts.</li> </ul>

## Table of Contents

<b>1. Introduction .....</b>	<b>7</b>
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
<b>2. Azure Infrastructure .....</b>	<b>8</b>
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).....	15
2.5. Storage Account and Service Principal Configuration in Azure .....	18
2.6. Adding Secrets in GitHub.....	20
2.7. Execute GitHub Action for IaC deployment .....	22
2.8. Manual Configuration Changes on Azure Platform.....	23
2.8.1. OAuth 2.0 Configuration for SDR UI Application.....	23
2.8.2. Key Vault .....	27
2.8.3. Default Group Creation in Azure Cosmos DB for MongoDB Account .....	31
<b>3. Resource Validation.....</b>	<b>32</b>
3.1. Low-Level Design Document .....	32
3.2. Virtual Network .....	33
3.3. Subnet .....	37
3.4. Delegated Subnet.....	38
3.5. Other Resources .....	40
<b>4. Application Code Deployment.....</b>	<b>41</b>
4.1 GitHub Secrets used in Workflow .....	41
4.2 Deploy the UI Application .....	42
4.3 Deploy Back-End API and Function App.....	44
4.4 Deployment Verification .....	47
<b>5. APIM Setup .....</b>	<b>51</b>
5.1. Developer Portal Configuration .....	61
5.1.1. Add App registration for Azure AD login .....	62
5.1.2. Identities in APIM .....	63
5.1.3. Change Publisher Name in APIM.....	64

**5.1.4. Add User to the AD Group for Developer Portal Access .....64**

## 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 .....	14
Figure 7 Create New User .....	15
Figure 8 Access Control (IAM) .....	16
Figure 9 Add Role Assignment.....	16
Figure 10 Select Roles.....	17
Figure 11 Select Members for Role Assignment .....	17
Figure 12 View Role Assignments.....	18
Figure 13 GitHub Actions Secrets .....	21
Figure 14 Add new GitHub Action Secret.....	22
Figure 15 Azure AD - App Registration .....	23
Figure 16 Azure AD - App Registration - Redirect URI.....	24
Figure 17 Azure AD - App Registration - Expose an API.....	24
Figure 18 Azure AD - App Registration - API Permission .....	25
Figure 19 Azure AD - App Registration - Add API Permission.....	25
Figure 20 Azure AD - App Registration - API Permission - Grant Admin Consent.....	26
Figure 21 Azure AD - App Registration - Certificates & Secrets .....	26
Figure 22 Azure AD - App Registration - Essential Secrets.....	27
Figure 23 Add Key Vault Access Policy.....	29
Figure 24 Select Secret Permissions in Access Policy in Key Vault .....	29
Figure 25 Select Principal (List of users) In Key Vault Access Policy .....	30
Figure 26 Create Secrets in Key Vault .....	30
Figure 27 Add Secrets values in Key Vault .....	31
Figure 29 Resource Naming Convention.....	32
Figure 30 SDR Resource Naming Convention .....	32
Figure 31 VNet Configurations .....	34
Figure 32 Virtual Network.....	35
Figure 33 Virtual Network - DDoS protection.....	35
Figure 34 Virtual Network - Tags.....	36
Figure 35 Virtual Network - Diagnostic Setting .....	36
Figure 36 Subnet Details.....	37
Figure 37 Delegated Subnet-001 Details.....	38
Figure 38 Delegated Subnet-001 Service Endpoints Details .....	39
Figure 39 Delegated Subnet-002 Details.....	39
Figure 40 Delegated Subnet-002 Service Endpoints Details .....	39
Figure 41 Function App Delegated Subnet-002 Details .....	40
Figure 42 Function App Delegated Subnet-002 Service Endpoints Details .....	40
Figure 43 SDR UI Repo - GitHub Actions.....	43
Figure 44 GitHub Actions - CI Workflow .....	43

---

Figure 45 GitHub - CI Workflow Run .....	44
Figure 46 GitHub CI Workflow Output .....	44
Figure 47 GitHub SDR API Repo .....	45
Figure 48 GitHub Actions CI Worklfow .....	45
Figure 49 GitHub CI Workflow Run .....	46
Figure 50 GitHub CI Workflow Run .....	46
Figure 51 GitHub CI Workflow Output .....	47
Figure 52 Azure Portal .....	47
Figure 53 Azure Portal - SDR UI App Service .....	48
Figure 54 SDR UI App Service - Deployment Center Details .....	48
Figure 55 SDR UI App Service Deployment Center -Logs .....	49
Figure 56 Azure Portal .....	49
Figure 57 Azure Portal - SDR Function App Service .....	50
Figure 58 SDR Function App Service – Deployment Center .....	50
Figure 59 SDR Function App Service Deployment Center – Logs.....	51
Figure 60 Client Certificate.....	52
Figure 61 APIM Certificates .....	52
Figure 62 APIM Certificates - Client Certificates.....	53
Figure 63 APIM Certificates - Client Certificate .....	53
Figure 64 : APIM Inbound Policy for All APIs .....	54
Figure 65 APIM Inbound Processing.....	55
Figure 66 : Update the Api-Key header name .....	58
Figure 67 APIM - Inbound Policy.....	59
Figure 68 : Enable Azure AD for Developer Portal .....	62
Figure 69 : API Permissions for APIM App Registration .....	63
Figure 70 : Add Sign-in tenant.....	63
Figure 71 : Redirect anonymous users to sign-in page.....	64
Figure 72 : Change Publisher Name .....	64
Figure 73 : Add User to Azure AD Group .....	65

## 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.

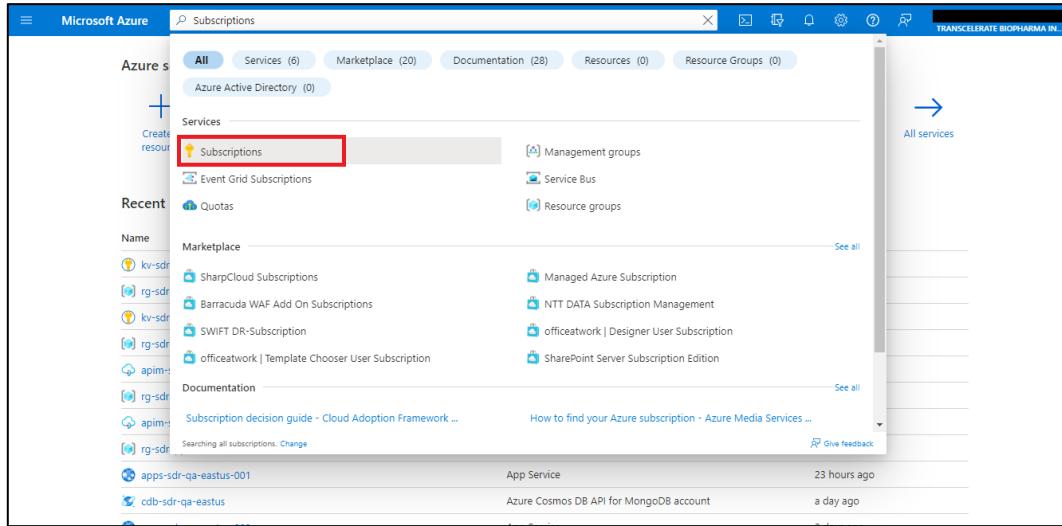
### STEPS:

- i. Sign into the Azure portal.

ii. On the Azure portal menu

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

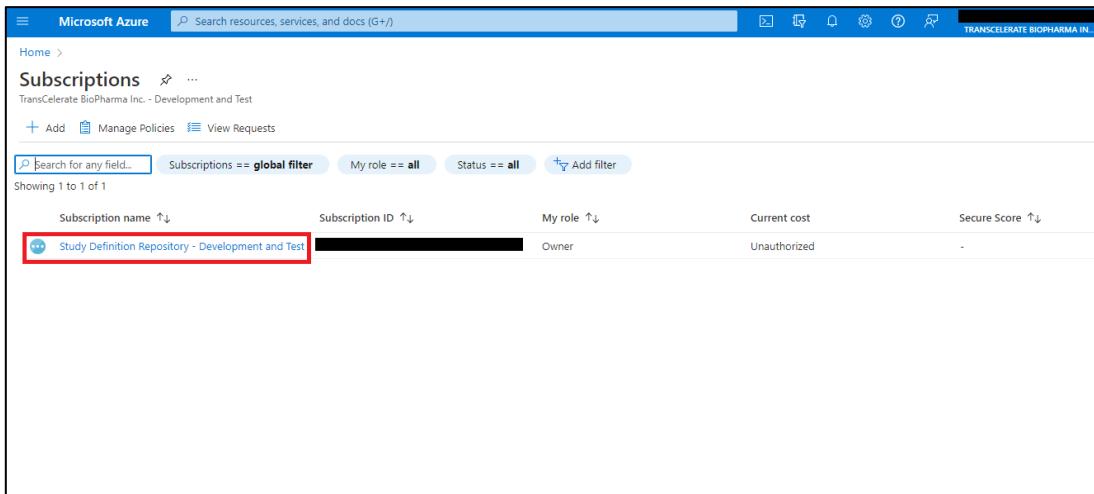
*Figure 1 Select Subscriptions in Azure Portal*



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	Managed Azure Subscription	23 hours ago
apim-rg-sdr	NTT DATA Subscription Management	23 hours ago
rg-sdr	officeatwork   Designer User Subscription	23 hours ago
rg-sdr	SharePoint Server Subscription Edition	23 hours ago
Subscription decision guide - Cloud Adoption Framework ...	How to find your Azure subscription - Azure Media Services ...	23 hours ago
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.

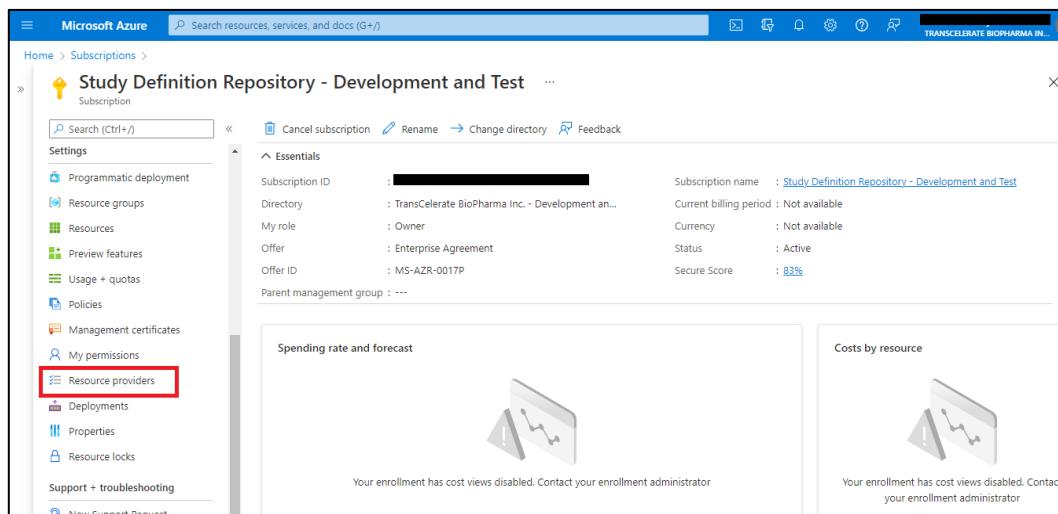
*Figure 2 Select Subscription*



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*

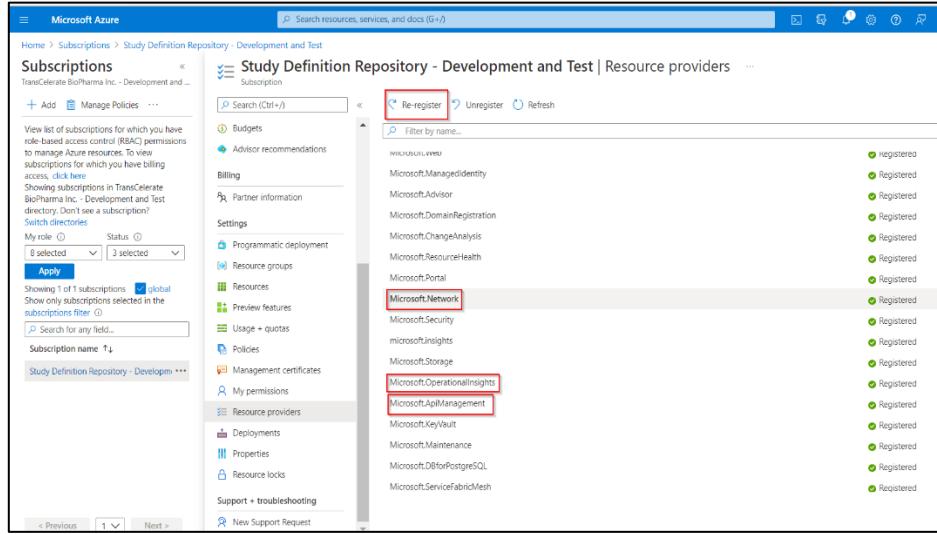


- v. Find the resource provider you want to register and select Register. To maintain least privileges in your subscription, only register the additional resource providers (other than default) that are required as listed below.

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.

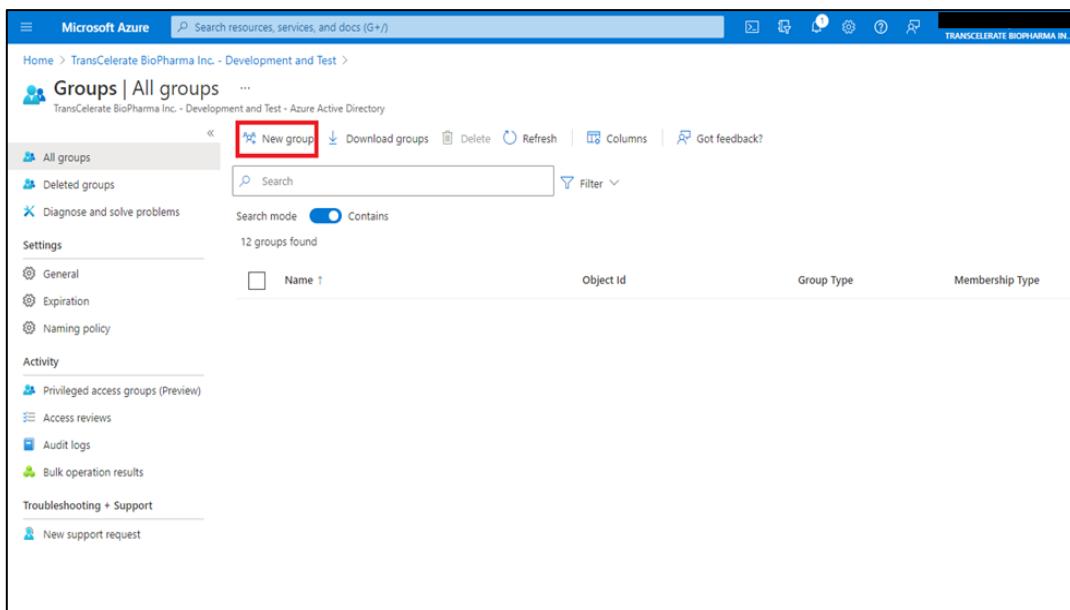
RBAC Group Name	Azure Built-In Role	Scope	Usage
SDR API Developer Portal Access		Azure Active Directory	Can access API Management developer portal and generate API key.

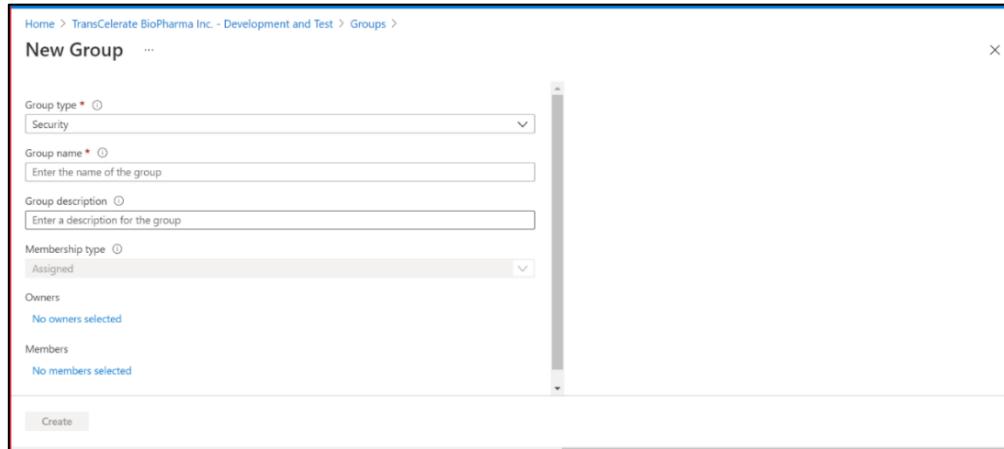
**Note:** To assign Azure AD roles to groups required Azure AD Premium P1 or P2 license, currently Azure AD Free plan has been leveraged. Follow this [documentation](#) to assign Azure AD role to groups.

#### 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



*Figure 6 Add New Group*

The screenshot shows a web-based application interface for creating a new group. At the top, there's a breadcrumb navigation: Home > TransCelerate BioPharma Inc. - Development and Test > Groups > New Group. The main area is titled "New Group". It contains several input fields:

- Group type: A dropdown menu showing "Security" is selected.
- Group name: An input field with placeholder text "Enter the name of the group".
- Group description: An input field with placeholder text "Enter a description for the group".
- Membership type: A dropdown menu showing "Assigned" is selected.
- Owners: A section indicating "No owners selected".
- Members: A section indicating "No members selected".

At the bottom right of the form is a "Create" button.

## 2.3. Create Users

### GOAL:

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

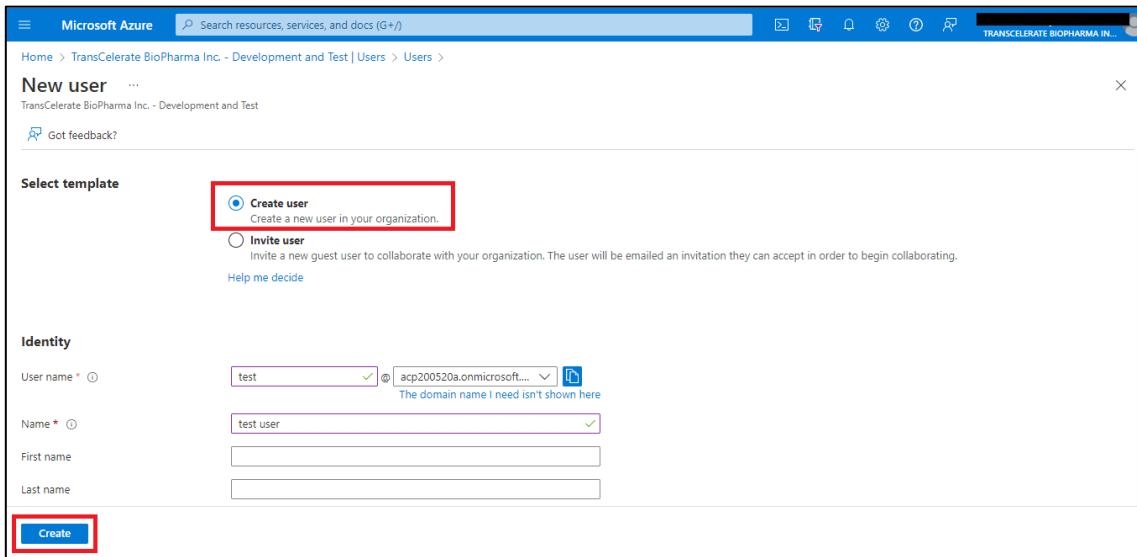
### PRE-REQUISITES:

- 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.

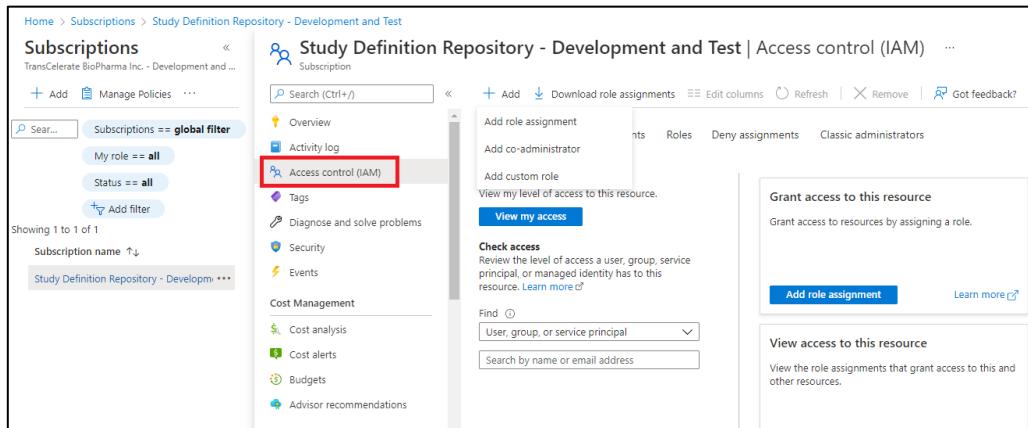
### STEPS:

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

At subscription level

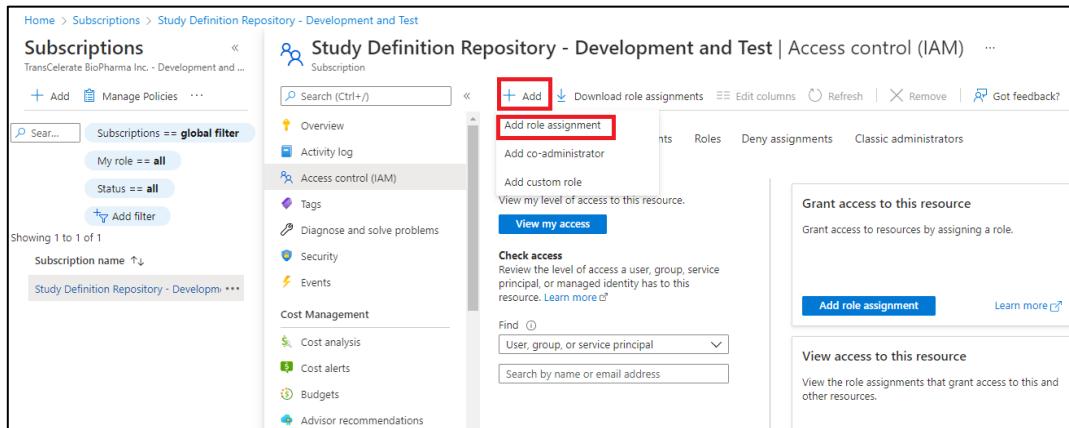
- Go to the subscription.
- On the left pane select Access control (IAM) as shown in below screenshot.

Figure 8 Access Control (IAM)



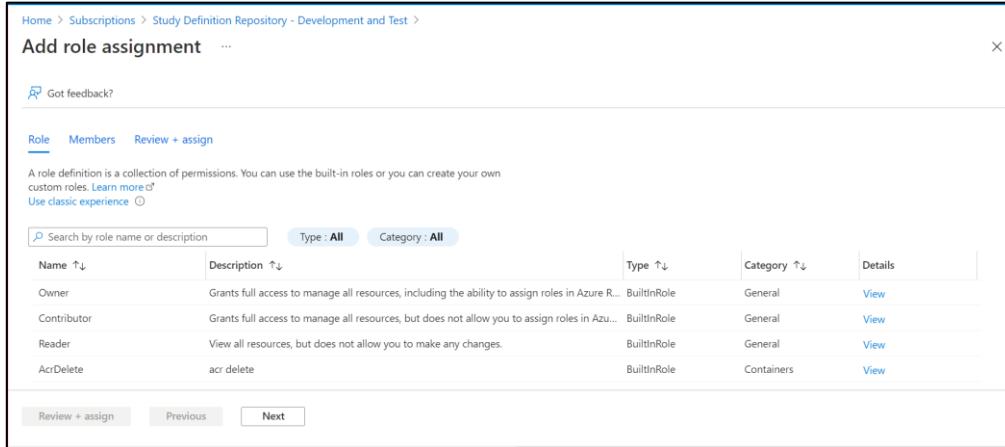
iii. Click on + Add and add the role assignment

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

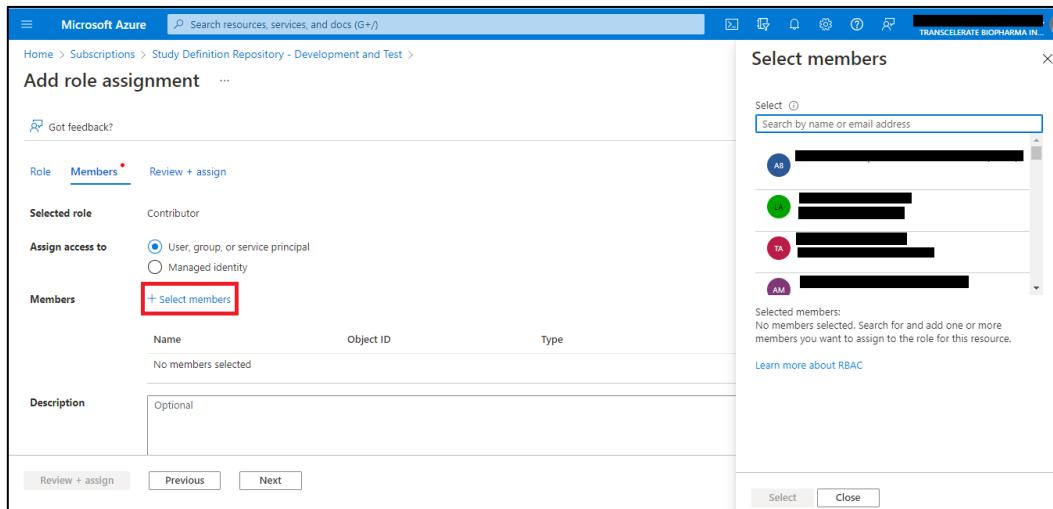


The screenshot shows the 'Add role assignment' interface. At the top, there's a breadcrumb navigation: Home > Subscriptions > Study Definition Repository - Development and Test > Add role assignment. Below the navigation, there's a 'Got feedback?' link and tabs for Role, Members, and Review + assign. The Role tab is selected. A note says: 'A role definition is a collection of permissions. You can use the built-in roles or you can create your own custom roles.' There are buttons for 'Search by role name or description', 'Type: All', and 'Category: All'. A table lists four built-in 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	<a href="#">View</a>
Contributor	Grants full access to manage all resources, but does not allow you to assign roles in Aze...	BuiltinRole	General	<a href="#">View</a>
Reader	View all resources, but does not allow you to make any changes.	BuiltinRole	General	<a href="#">View</a>
AcDelete	ac delete	BuiltinRole	Containers	<a href="#">View</a>

At the bottom, there are 'Review + assign', 'Previous', and 'Next' buttons.

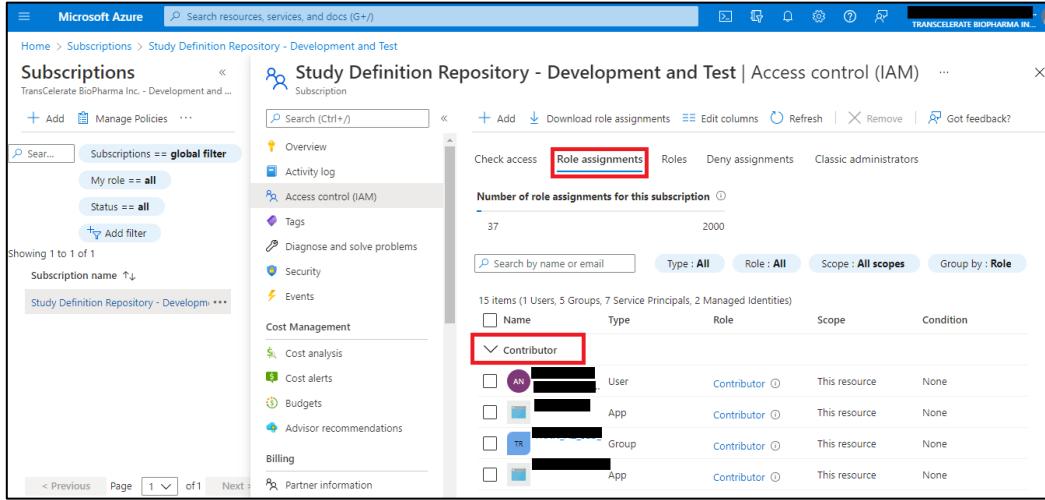
Figure 11 Select Members for Role Assignment



The screenshot shows the 'Add role assignment' interface with the 'Members' tab selected. The 'Selected role' is set to 'Contributor'. Under 'Assign access to', the radio button for 'User, group, or service principal' is selected. The 'Members' section has a red box around the '+ Select members' button. A modal window titled 'Select members' is open, showing a search bar and a list of users with colored icons (blue, green, red, purple) and blacked-out names. Below the list, it says 'Selected members: No members selected. Search for and add one or more members you want to assign to the role for this resource.' There are 'Select' and 'Close' buttons at the bottom of the modal.

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

Figure 12 View Role Assignments



Name	Type	Role	Scope	Condition
AN [REDACTED]	User	Contributor	This resource	None
[REDACTED]	App	Contributor	This resource	None
TS [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).

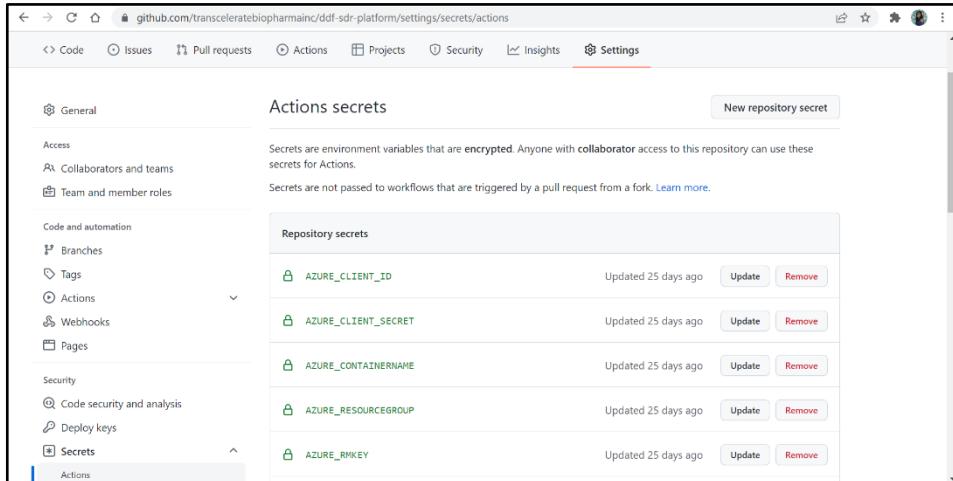
Secret Name	Values
ADgroup3	Provide the name of the Azure AD Group for Reader access on App & Core Resource Groups.
ADGROUP_DEVELOPER_PORTAL	External Id of the AD group for access to the APIM developer portal. e.g.: aad://abc123456a.onmicrosoft.com/groups/e12b0c1ea11a-12b1-123f-3e4f11111a20
Serviceprincipal	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.

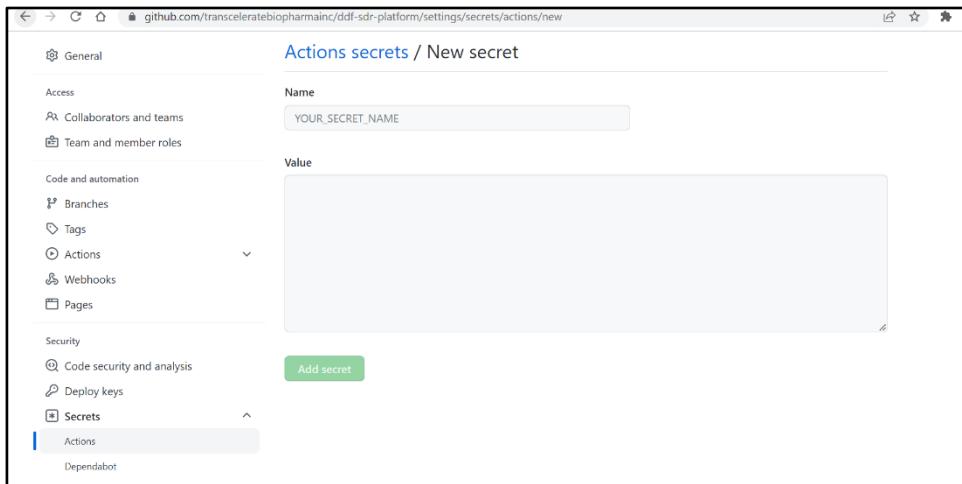
- i. Go to repository settings.
- ii. On the lower left-hand side of the screen, click on Secrets.
- iii. Under that click on Actions.
- iv. Then click on New Repository Secret.

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:**

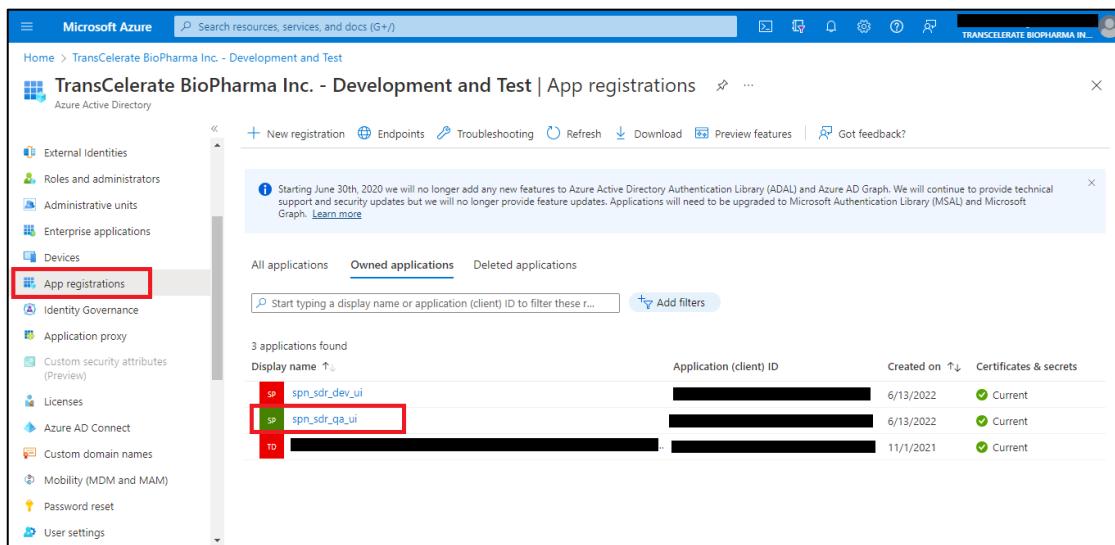
- Global Administrator level of access at Active Directory level.

**STEPS**

**SDR UI APP REGISTRATION:**

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

Figure 15 Azure AD - App Registration

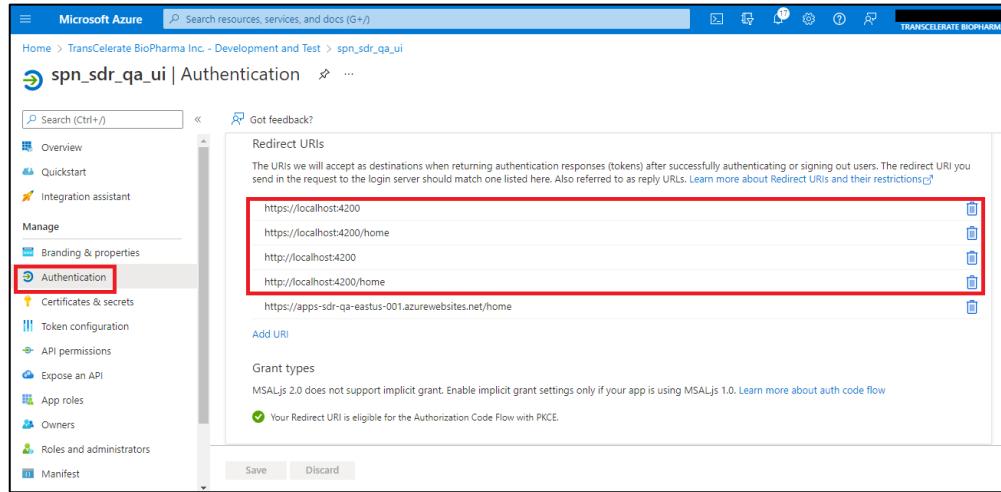


The screenshot shows the Microsoft Azure portal interface for 'App registrations'. The left sidebar has a red box around the 'App registrations' option under 'Azure Active Directory'. The main area shows a table of registered applications:

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

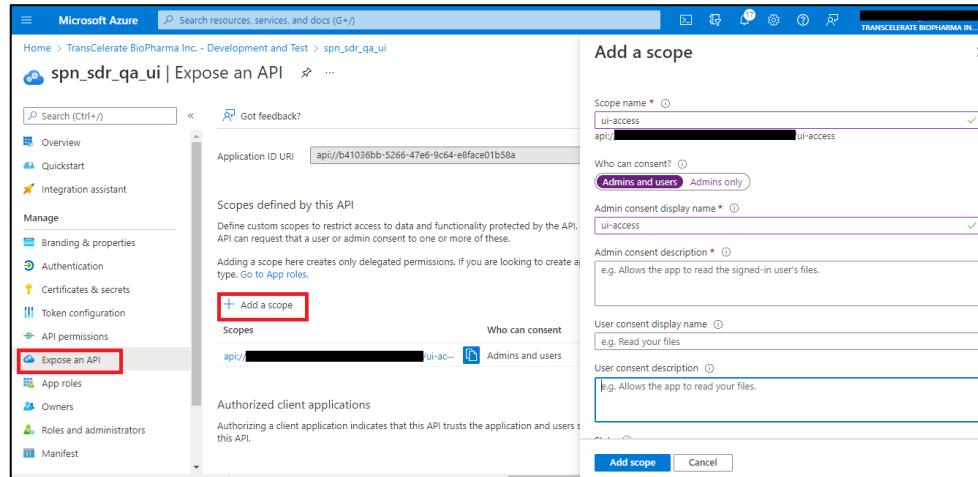
- Go to Authentication blade, add additional Redirect URL as needed. Add localhost URL's for testing the application from development machine.

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

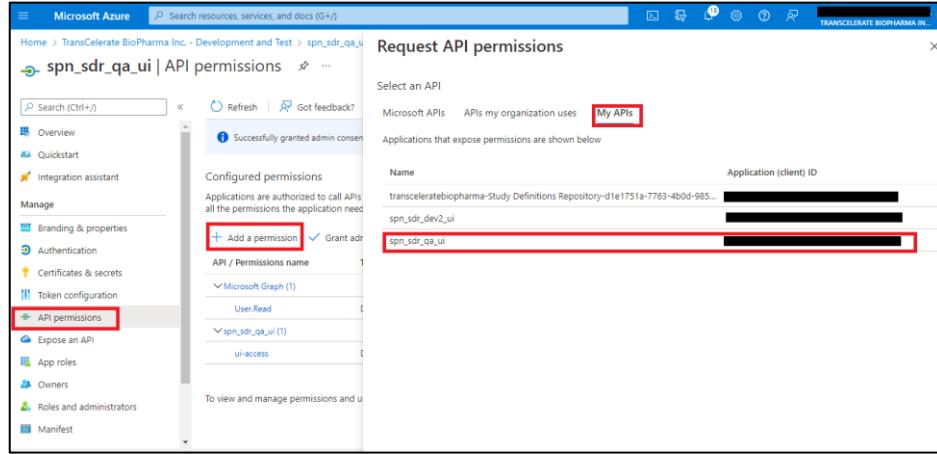
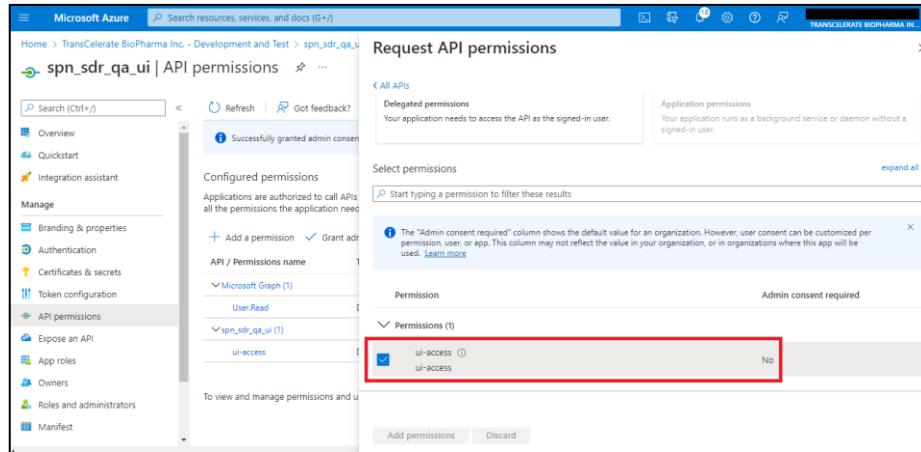
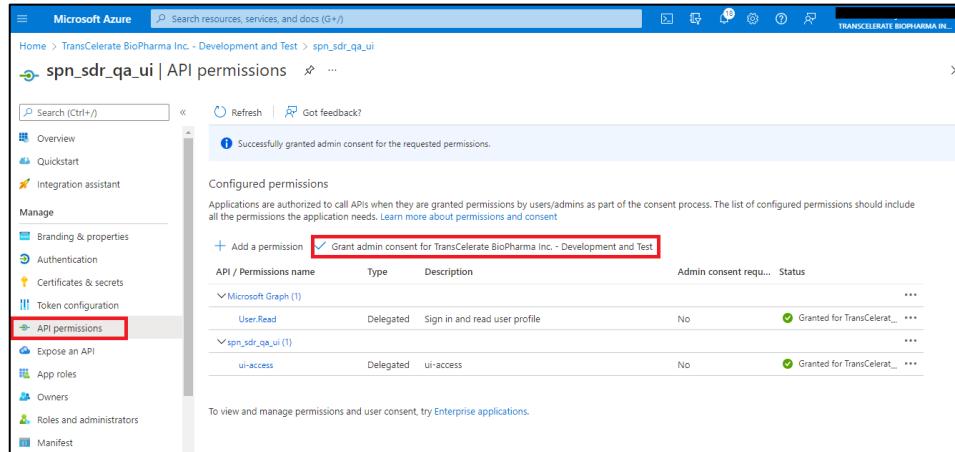


Figure 19 Azure AD - App Registration - Add API Permission



## v. Grant Admin consent.

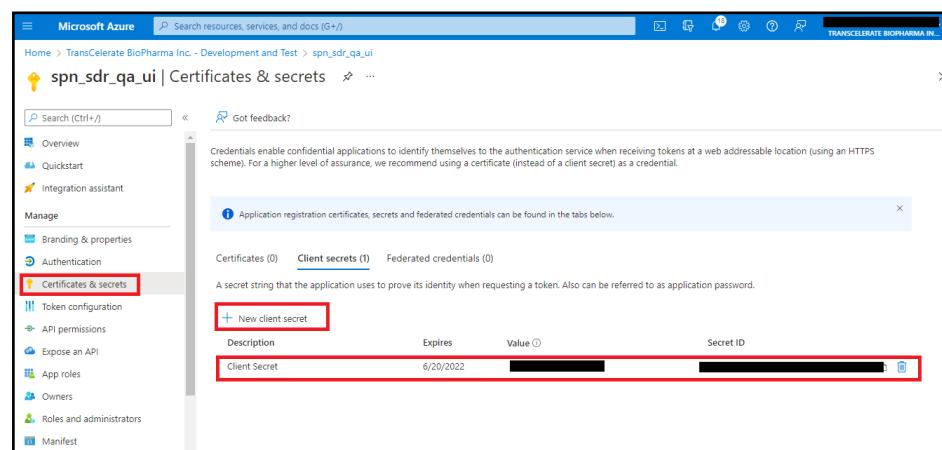
Figure 20 Azure AD - App Registration - API Permission - Grant Admin Consent



The screenshot shows the Microsoft Azure portal interface for app registration. The left sidebar lists various management options like Overview, Quickstart, Integration assistant, etc. Under the 'Manage' section, 'API permissions' is selected and highlighted with a red box. On the right, a message says 'Successfully granted admin consent for the requested permissions.' Below it, a table titled 'Configured permissions' shows two entries: 'User.Read' and 'ui-access', both of which have their 'Admin consent required' status set to 'No' and are marked as 'Granted for TransCelerate...'.

- 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

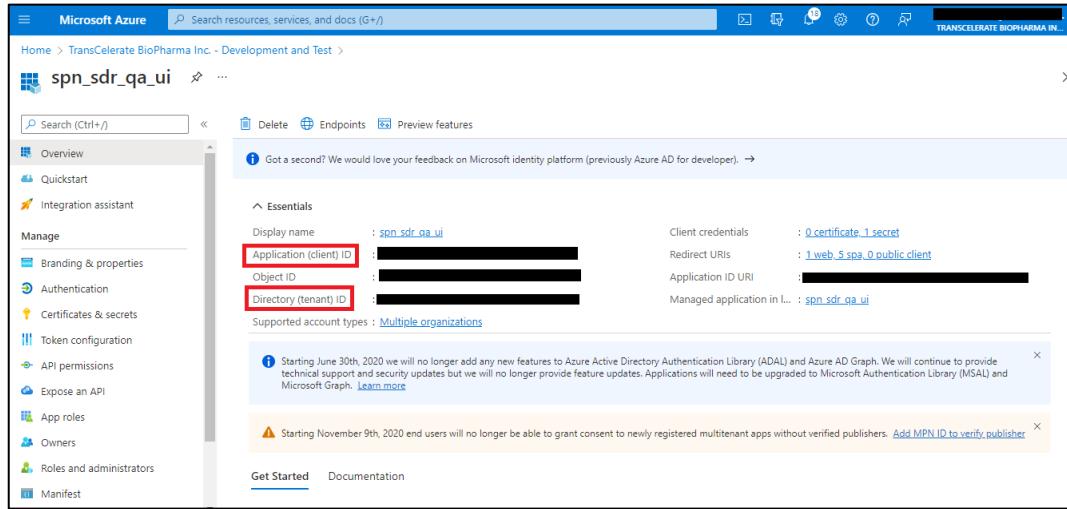


The screenshot shows the Microsoft Azure portal interface for app registration under the 'Certificates & secrets' tab. A red box highlights the 'New client secret' button. The table below shows one entry: 'Client Secret' with an expiration date of '6/20/2022'. The 'Value' column is redacted with a black box.

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
<b>Apim-BaseUrl</b>	Provide API Management URL as key-value (E.g., https://apim-sdr-qa-eastus.azure-api.net/studydefinitionrepository/v1/)
<b>ApplicationInsights--InstrumentationKey</b>	Provide Application Insights Instrumentation key
<b>AzureAd-Audience</b>	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)

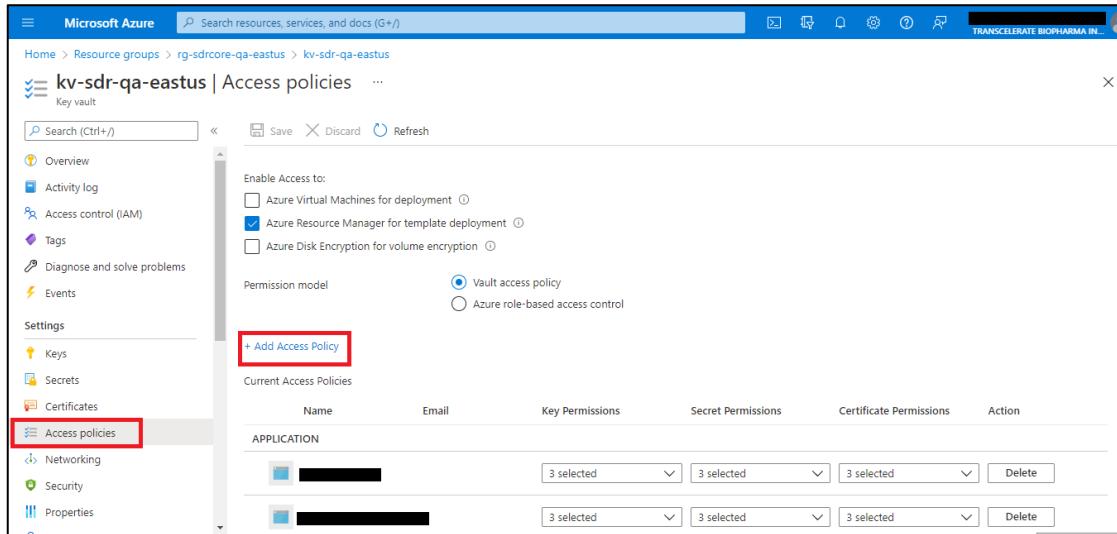
Secret Name	Secret Value
AzureAd--Audience	Provide the UI app registration Application ID URI (E.g.: api://0000-0000-0000-0000)
AzureAd-ClientSecret	Provide App Registration Client secret value.
AppInsights-ApiKey	Provide Application Insights Api Key Value
AppInsights-AppId	Provide Application Insights AppId
AppInsights-RESTApiUrl	Provide Default URL <a href="https://api.applicationinsights.io/v1/apps">https://api.applicationinsights.io/v1/apps</a>
AzureAd-Authority	Provide the Azure Ad authority value ( <a href="https://login.microsoftonline.com/">https://login.microsoftonline.com/</a> (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.: <a href="https://Front End App service URL (UI)/home">https://Front End App service URL (UI)/home</a> )
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 : <a href="#">ddf-sdr-api-usdm-version-mapping</a>
SdrCptMasterDataMapping	Copy JSON content from file : <a href="#">ddf-sdr-cpt-master-data-mapping</a>
AzureServiceBusConnectionString	Provide the Azure Service Bus Connection String

### 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

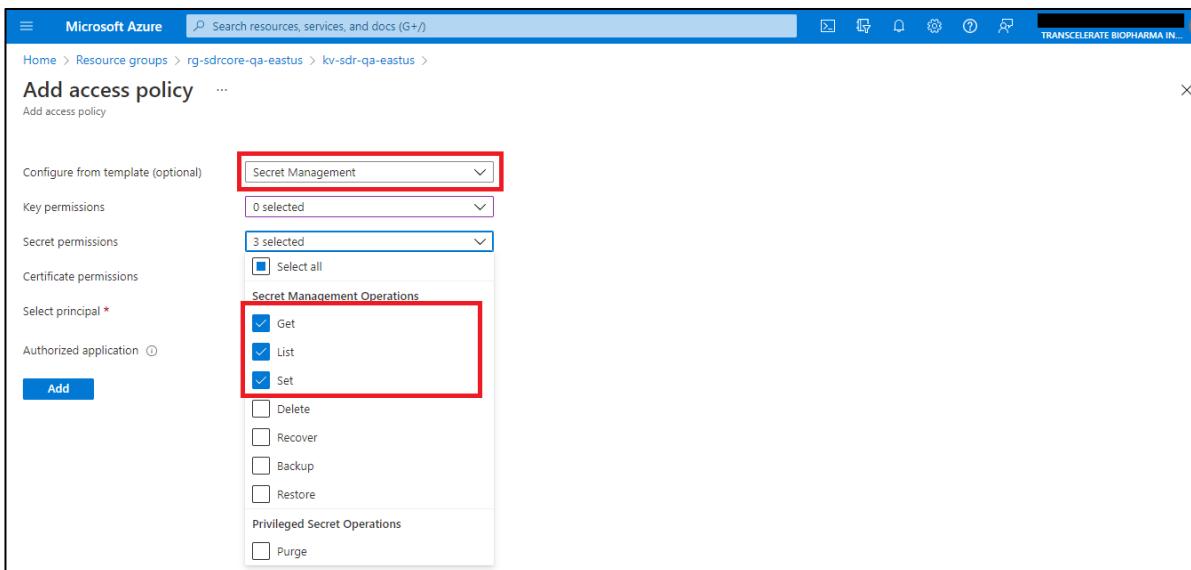
*Figure 23 Add Key Vault Access Policy*



The screenshot shows the Microsoft Azure portal interface for managing access policies in a Key Vault. The left sidebar has 'Access policies' selected. The main area shows 'Enable Access to:' options (Azure VM deployment, ARM template deployment, Disk Encryption) and a 'Permission model' section (Vault access policy selected). A red box highlights the '+ Add Access Policy' button. Below it, a table lists current access policies under the 'APPLICATION' tab, with two entries showing columns for Name, Email, Key Permissions, Secret Permissions, Certificate Permissions, and Action.

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

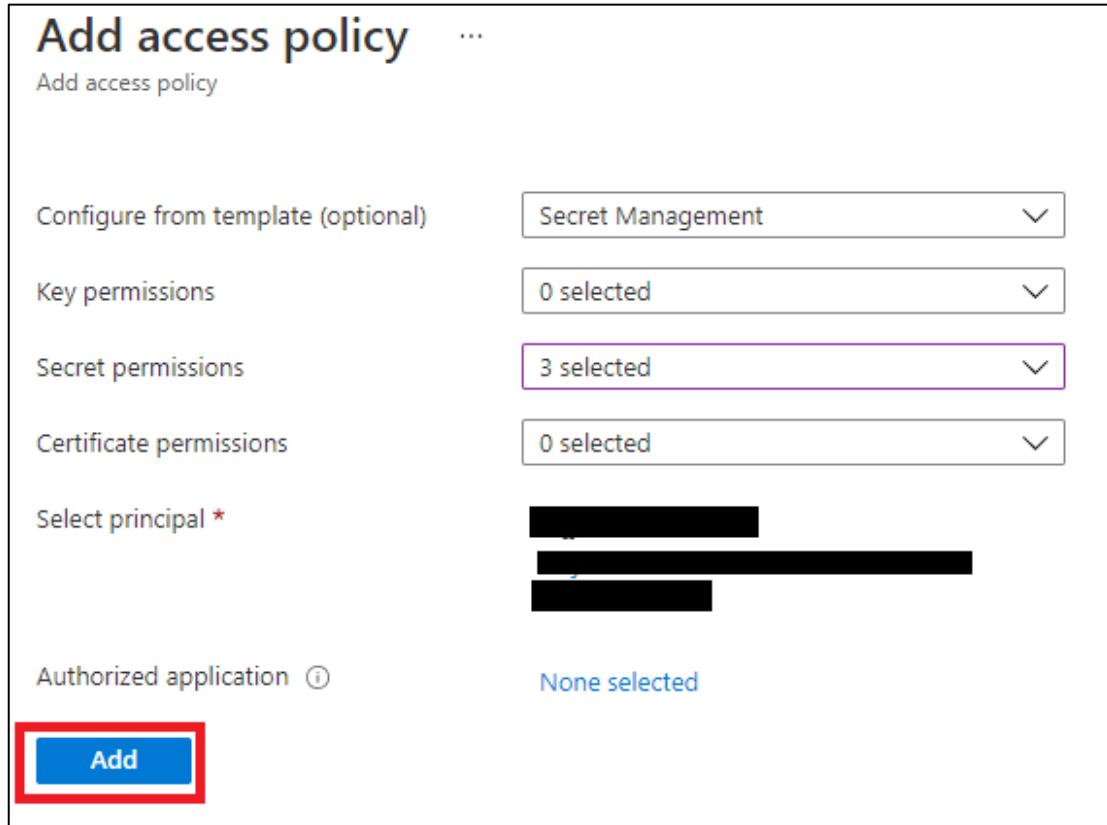
*Figure 24 Select Secret Permissions in Access Policy in Key Vault*



The screenshot shows the 'Add access policy' dialog. Under 'Secret permissions', 'Secret Management' is selected in a dropdown menu. In the 'Secret Management Operations' section, three checkboxes are checked: 'Get', 'List', and 'Set'. Other options like 'Delete', 'Recover', 'Backup', 'Restore', and 'Privileged Secret Operations' are also listed but not selected. A red box highlights the 'Secret Management' dropdown and the 'Secret Management Operations' section.

- 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



**Add access policy**

Add access policy

Configure from template (optional)

Secret Management

Key permissions

0 selected

Secret permissions

3 selected

Certificate permissions

0 selected

Select principal \*

Authorized application ⓘ

None selected

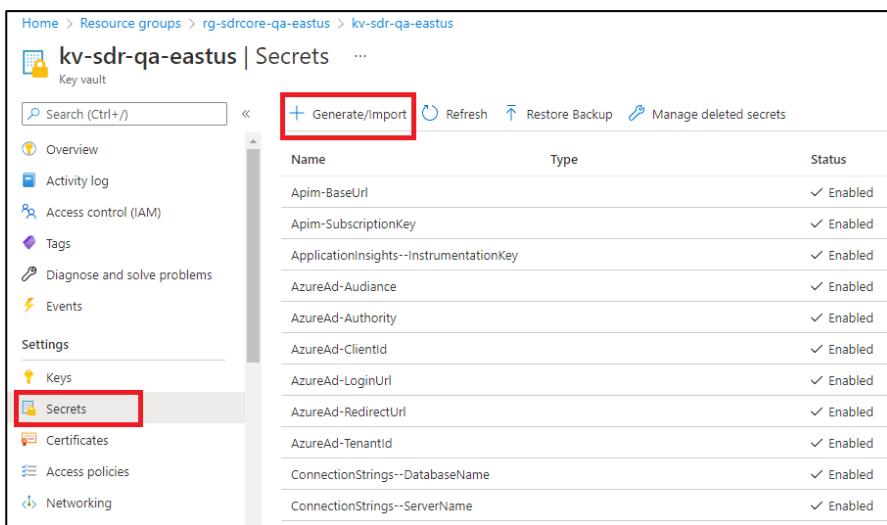
**Add**

#### STEPS FOR ADDING KEY VAULT SECRETS:

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

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

Figure 26 Create Secrets in Key Vault

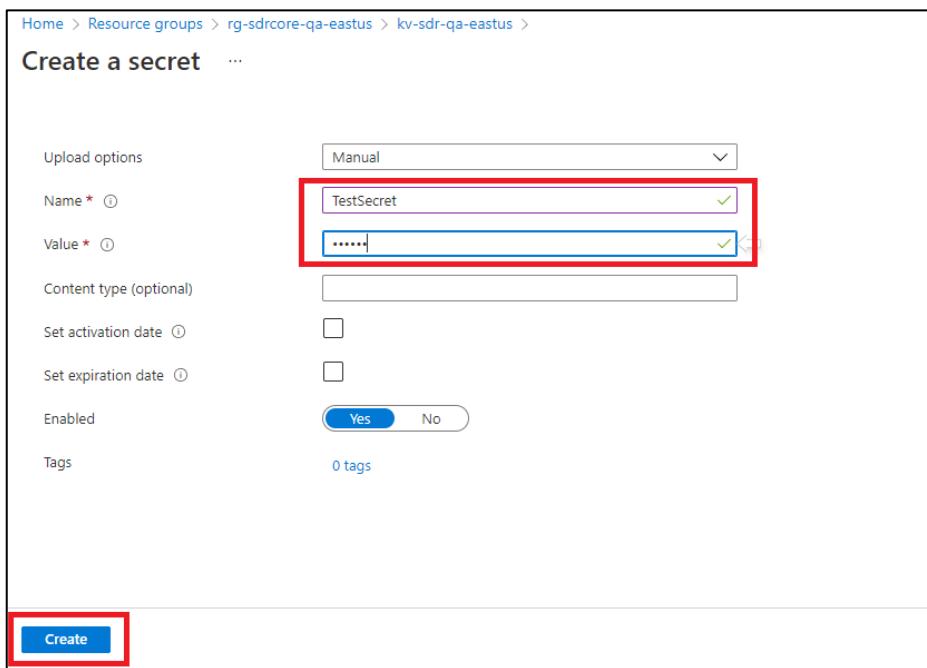


kv-sdr-qa-eastus | Secrets

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' page in the Azure portal. The 'Name' field is populated with 'TestSecret' and the 'Value' field contains '.....'. Both fields are highlighted with a red box. The 'Create' button at the bottom left is also highlighted with a red box.

### 2.8.3. Default Group Creation in Azure Cosmos DB for MongoDB Account

**GOAL:**

- Add default group in Groups Collection.

**PRE-REQUISITES:**

- Groups Collection must have been created.
- The IP of the user must be whitelisted in the Azure Cosmos DB for MongoDB Account.

**STEPS TO ADD DEFAULT GROUP:**

- i. Open Studio3T or any equivalent tool to connect to MongoDB and connect to the Azure Cosmos DB for MongoDB Account using the connection string.
- ii. Then execute the below command to insert a new Group in the Groups collection.

```
db.Groups.insertOne({ SDRGroups: [ { groupId: '4aabb043-4430-4539-8694-fbbc1bbab3ad', groupName: 'Def.Group', groupDescription: 'Detailed description of group', permission: 'READ_WRITE', groupFilter: [ { groupFieldName: 'studyType', groupFilterValues: [ { id: 'ALL', title: 'ALL' } ] } ], users: null, groupCreatedBy: 'admin', groupCreatedOn: ISODate('2023-12-20T01:00:00Z'), groupModifiedBy: 'admin', groupModifiedOn: ISODate('2023-12-20T01:00:00Z'), groupEnabled: true } ] })
```

**Note:** Mention current date time for **groupCreatedOn** and **groupModifiedOn** fields in the script before executing.

### 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 28 Resource Naming Convention

<b>Naming Convention</b>	<pre>&lt;ResourceType&gt;-&lt;App/Svc&gt;-&lt;Purpose/Segment/Environment&gt;-&lt;region&gt;- ###</pre> <hr/> <pre>&lt;ResourceType&gt; = Mandatory (upto 4 Characters) &lt;App/Svc&gt; = Optional (upto 5 Characters) &lt;Purpose/Segment/Environment&gt;= Mandatory (Unlimited) &lt;Region&gt; = Optional (upto 6 Characters) &lt;###&gt; = Optional (Instance Number or level # = 3 Characters)</pre>				

Figure 29 SDR Resource Naming Convention

A	B	C
1 Resource Naming Convention	Dev	
2 Resource Group1	rg-sdrcore-dev-eastus	
3 Resource Group2	rg-sdrapp-dev-eastus	
4 Vnet	vnet-sdr-dev-eastus	
5 Subnet	snet-sdr-dev-eastus	
6 Delegated Subnet1	dsnet-sdrui-dev-eastus-001	
7 Delegated Subnet2	dsnet-sdrapi-dev-eastus-002	
8 Delegated Subnet3	dsnetfunapp-sdrapi-dev-eastus-002	
9 App Service1	apps-sdrui-dev-eastus-001	
10 App Service2	apps-sdrapi-dev-eastus-002	
11 Azure Function App	funapp-sdrapi-dev-eastus	
12 Azure Service Bus	asb-sdrapi-dev-eastus	
13 App Service Plan1	asp-sdrui-dev-eastus-001	
14 App Service Plan2	asp-sdrapi-dev-eastus-002	
15 App Service Plan3	funasp-sdrapi-dev-eastus-003	
16 Azure Container Registry	acrsdrdeveastus	
17 API Management	apim-sdr-dev-eastus	
18 Application Insights	appin-sdr-dev-eastus	
19 CosmosDB	cdb-sdr-dev-eastus	
20 Log Analytics Workspace	law-sdr-dev-eastus	
21 Key Vault	kv-sdr-dev-eastus	
22 Storage Account	sasdrdeveastus	
23 Terraform State File	tfstatefiledev	
24 Diagnostic Setting Name	diags-vnet-sdr-dev-eastus	
25 Network Security Group	nsg-sdr-dev-eastus	
26 Public IP	pip-sdr-dev-eastus	
27 Private endpoint	pe-sdr-dev-eastus	
28		
29		
30		
<span style="border: 1px solid green; padding: 2px;">Resource Naming Convention</span> <span style="border: 1px solid green; padding: 2px;">Resource Group</span> <span style="border: 1px solid green; padding: 2px;">Network S</span>		

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

### 3.2. Virtual Network

Validation of Virtual Network (VNet) configuration

Figure 30 VNet Configurations

Vnet				EnvName
Basics	Project details	Subscription		Subscription Name
		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 Settings	Diagnostic setting name			diags-vnet-SubName-EnvName-eastus
	Logs	Category group	allLogs	Disable
		Categories	VMProtection Alerts	Disable
	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

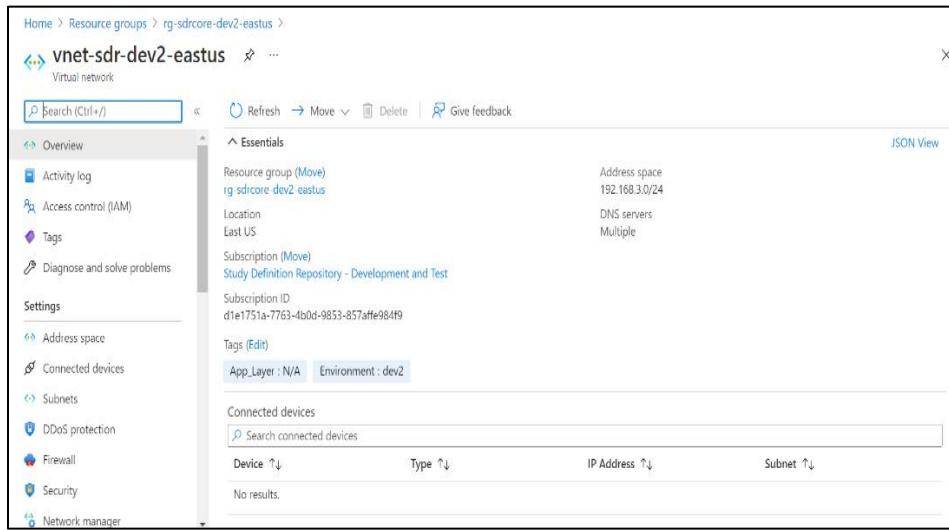
## PRE-REQUISITES:

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

## STEPS

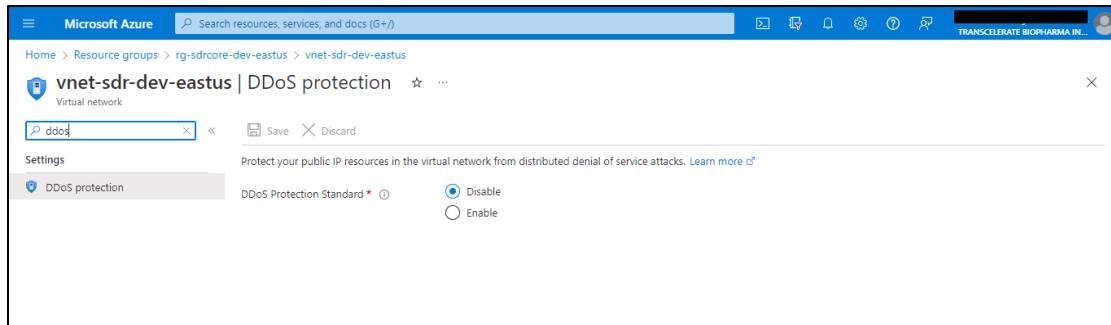
- i. Login to Azure Portal
- ii. Click on the Resource Groups tab.
- iii. Select the Resource group for VNet configuration.
- iv. Verify that the Basic details like Subscription, Resource Group, Name and Region is as per the LLD
- v. Verify that the IPv4 Address Space is as per the LLD

Figure 31 Virtual Network



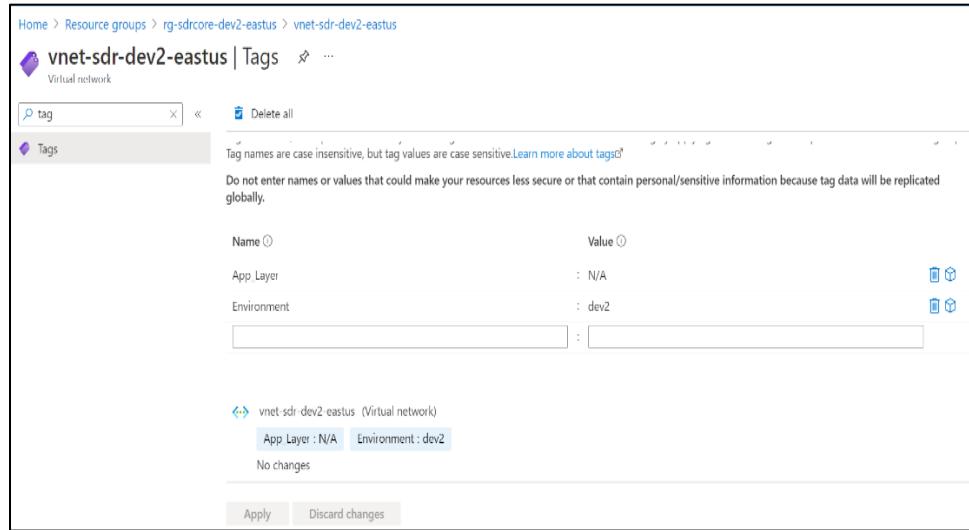
- vi. Go to Security tab and verify that Bastion Host is set as per the LLD.
- vii. Go to DDoS Protection tab and verify that it is set as per the LLD.

Figure 32 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.

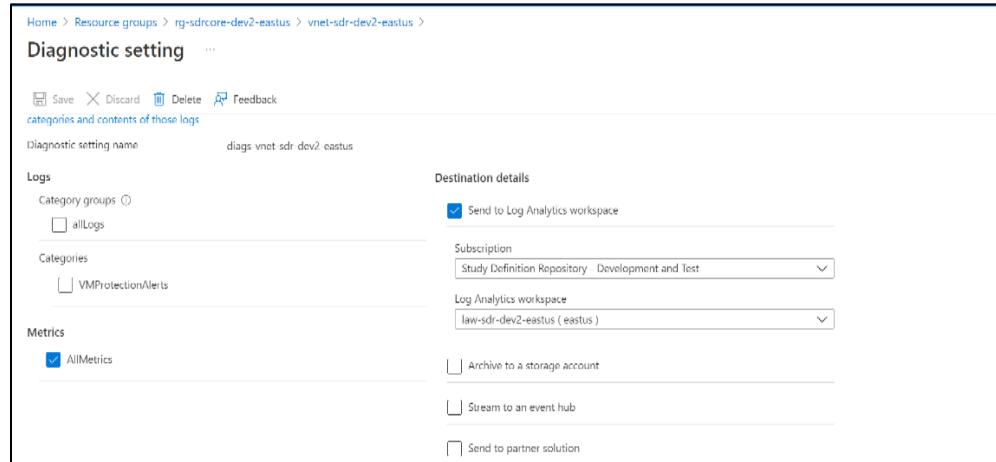
Figure 33 Virtual Network - Tags



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 34 Virtual Network - Diagnostic Setting



### 3.3. Subnet

Validation of Virtual Subnet configuration.

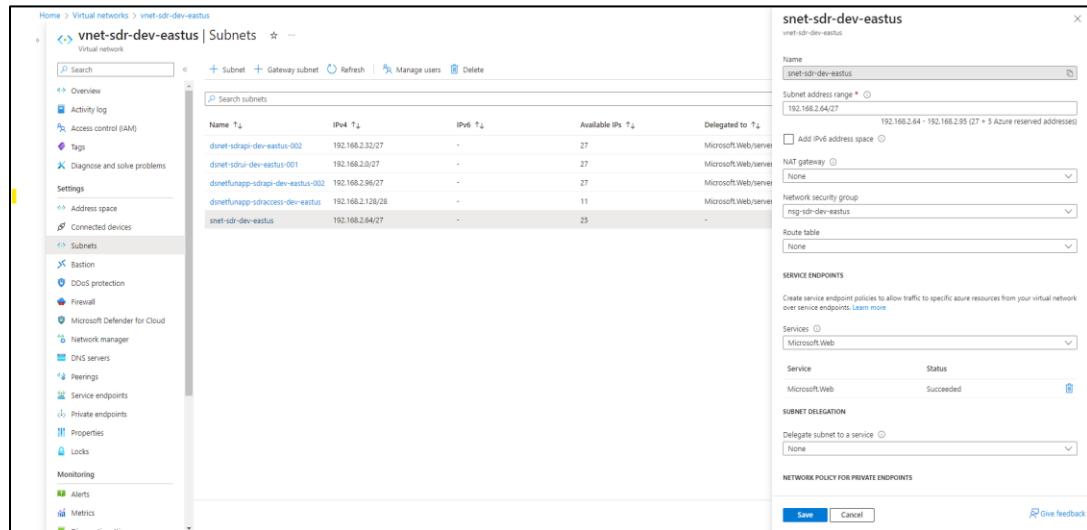
#### PRE-REQUISITES:

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

#### STEPS:

- Login to Azure Portal
- Click on the Resource Groups tab.
- Select the Resource group for VNet configuration.
- 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 35 Subnet Details



Name	IPv4	IPv6	Available IPs	Delegated to
dnet-sdrapi-dev-eastus-002	192.168.2.32/27	-	27	Microsoft/Web/server
dnet-sdrui-dev-eastus-001	192.168.2.0/27	-	27	Microsoft/Web/server
dnetfunapp-sdraccess-dev-eastus	192.168.2.96/27	-	11	Microsoft/Web/server
snet-sdr-dev-eastus	192.168.2.64/27	-	25	-

### 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 36 Delegated Subnet-001 Details

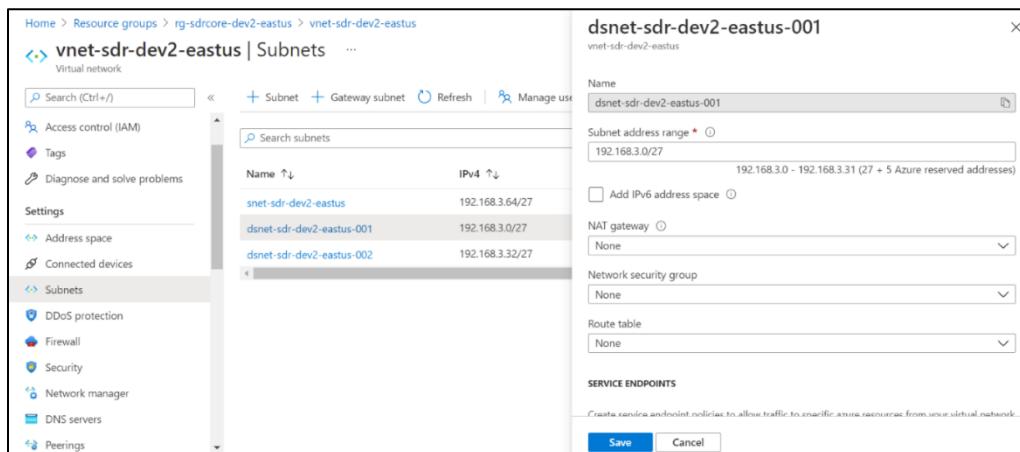
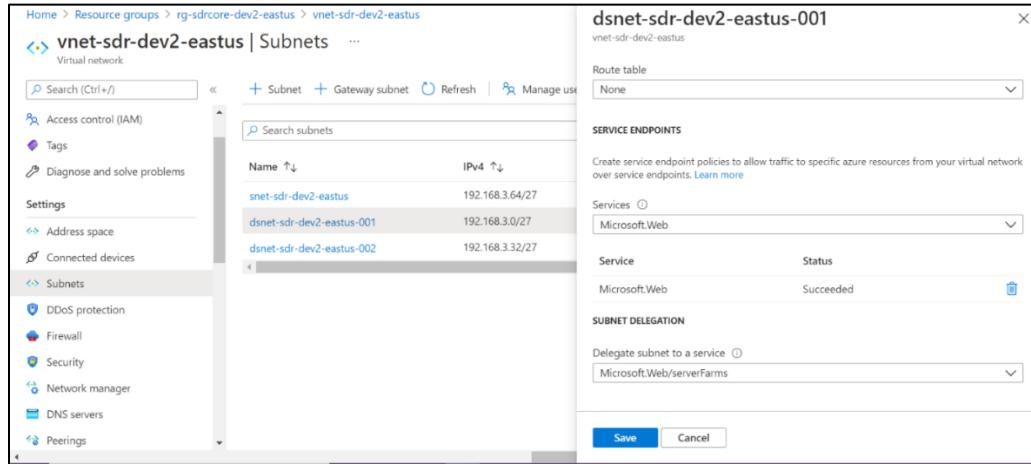


Figure 37 Delegated Subnet-001 Service Endpoints Details



Name	IPv4
snet-sdr-dev2-eastus	192.168.3.64/27
dsnet-sdr-dev2-eastus-001	192.168.3.0/27
dsnet-sdr-dev2-eastus-002	192.168.3.32/27

**SERVICE ENDPOINTS**

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

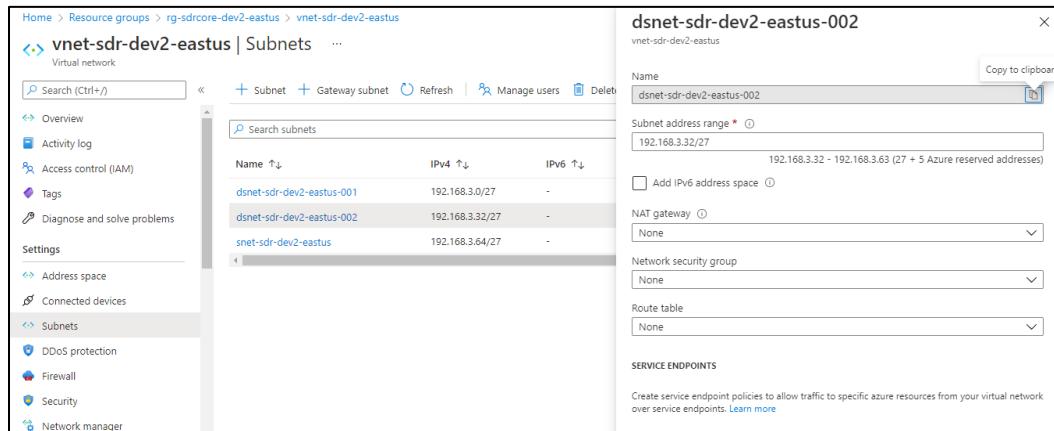
Service	Status
Microsoft.Web	Succeeded

**SUBNET DELEGATION**

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

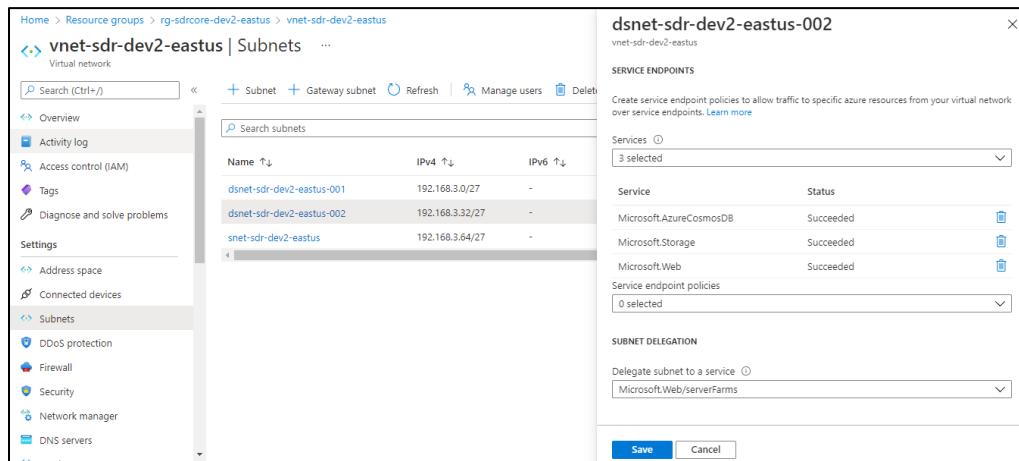
Microsoft.Web/serverFarms
---------------------------

Figure 38 Delegated Subnet-002 Details



Name	IPv4	IPv6
dsnet-sdr-dev2-eastus-001	192.168.3.0/27	-
dsnet-sdr-dev2-eastus-002	192.168.3.32/27	-
snet-sdr-dev2-eastus	192.168.3.64/27	-

Figure 39 Delegated Subnet-002 Service Endpoints Details



Name	IPv4	IPv6
dsnet-sdr-dev2-eastus-001	192.168.3.0/27	-
dsnet-sdr-dev2-eastus-002	192.168.3.32/27	-
snet-sdr-dev2-eastus	192.168.3.64/27	-

**SERVICE ENDPOINTS**

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

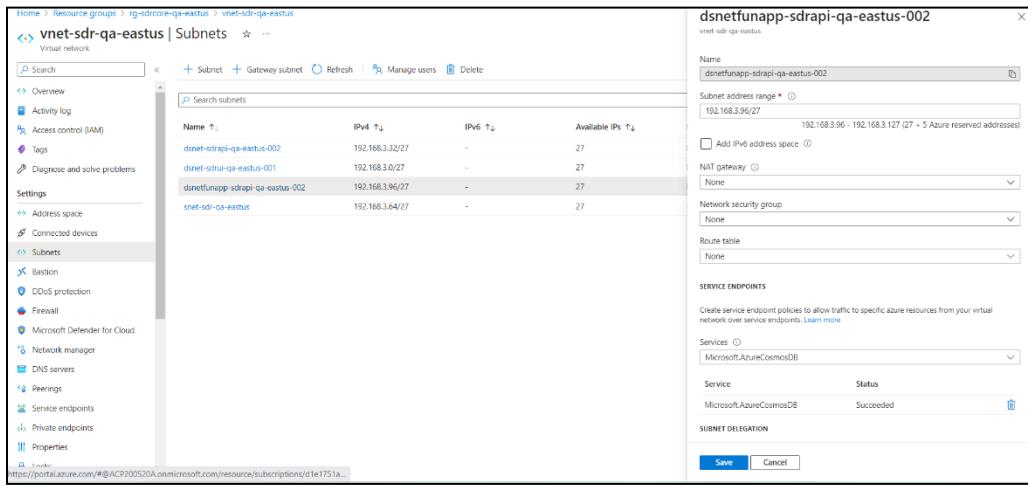
Service	Status
Microsoft.AzureCosmosDB	Succeeded
Microsoft.Storage	Succeeded
Microsoft.Web	Succeeded

**SUBNET DELEGATION**

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

Microsoft.Web/serverFarms
---------------------------

Figure 40 Function App Delegated Subnet-002 Details

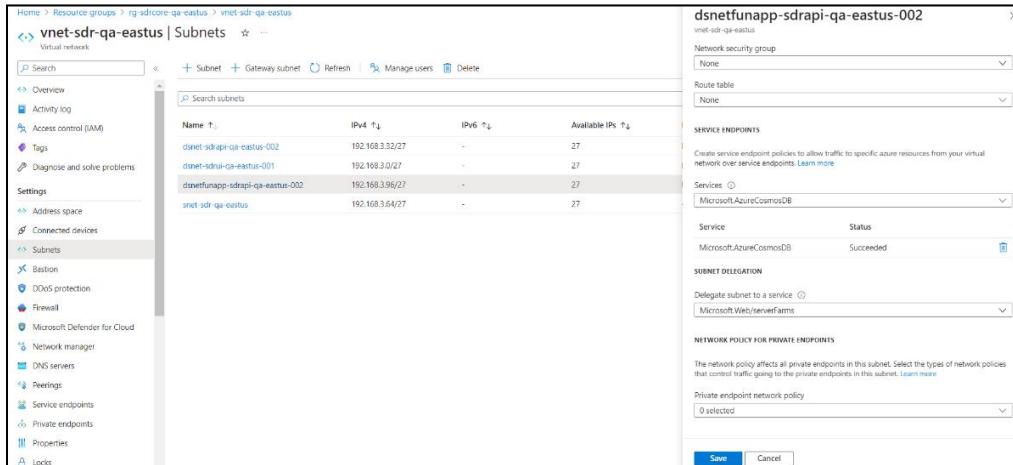


Name	IPv4	IPv6	Available IPs
dinet-sdrapi-qa-eastus-002	192.168.3.32/27	-	27
dinet-sdrui-qa-eastus-001	192.168.3.0/27	-	27
dsnetfunapp-sdrapi-qa-eastus-002	192.168.3.96/27	-	27
snet-sdr-qa-eastus	192.168.3.64/27	-	27

**dsnetfunapp-sdrapi-qa-eastus-002**

Name: dsnetfunapp-sdrapi-qa-eastus-002  
Subnet address range: 192.168.3.96 - 192.168.3.127 (27 + 5 Azure reserved addresses)  
Add IPv6 address space:   
NAT gateway: None  
Network security group: None  
Route table: None  
Service endpoints:  
Service: Microsoft.AzureCosmosDB  
Status: Succeeded  
SUBNET DELEGATION:  
Delegate to service: Microsoft.Web/serverFarms  
Save | Cancel

Figure 41 Function App Delegated Subnet-002 Service Endpoints Details



Name	IPv4	IPv6	Available IPs
dinet-sdrapi-qa-eastus-002	192.168.3.32/27	-	27
dinet-sdrui-qa-eastus-001	192.168.3.0/27	-	27
dsnetfunapp-sdrapi-qa-eastus-002	192.168.3.96/27	-	27
snet-sdr-qa-eastus	192.168.3.64/27	-	27

**dsnetfunapp-sdrapi-qa-eastus-002**

Network security group: None  
Route table: None  
Service endpoints:  
Service: Microsoft.AzureCosmosDB  
Status: Succeeded  
SUBNET DELEGATION:  
Delegate subnet to a service: Microsoft.Web/serverFarms  
Save | Cancel

**NETWORK POLICY FOR PRIVATE ENDPOINTS**  
The network policy affects all private endpoints in this subnet. Select the types of network policies that control traffic going to the private endpoints in this subnet. Learn more  
Private endpoint network policy: 0 selected

### 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
- Network Security Group
- Public IP
- App Services
- App Service Plans

- API Management
- Application Insights
- Cosmos DB
- Private Endpoint
- 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 and Key Vault name from Overview blade.
- For KEYVAULT\_NAME secret, consider Key Vault URL for API repo and Key Vault name for UI repo.
- 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

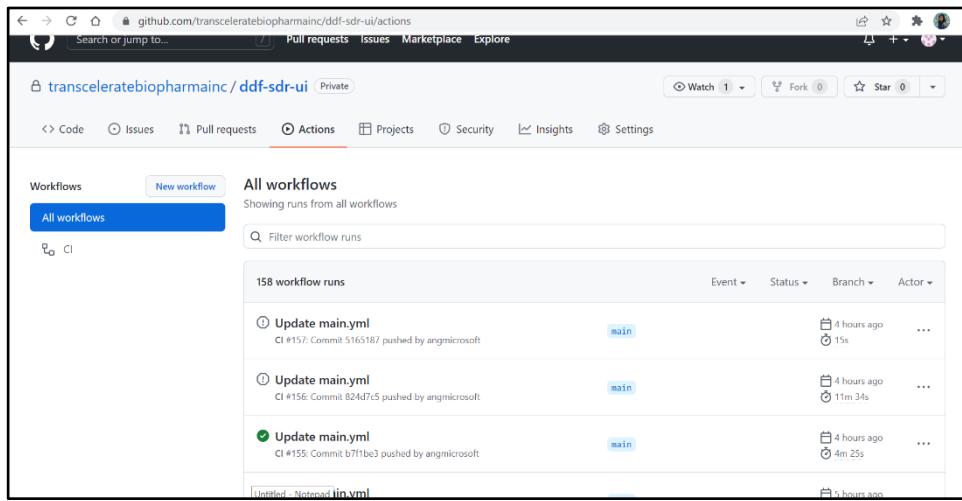
### PRE-REQUISITES

- Contributor level of access at Resource Group level.

### DEPLOYMENT STEPS:

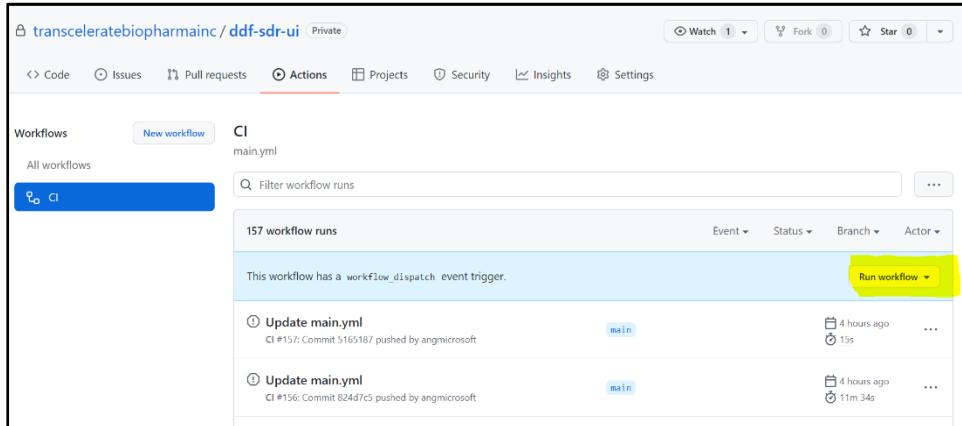
- i. Go to UI repository on GitHub.
- ii. Click on Actions tab.

Figure 42 SDR UI Repo - GitHub Actions



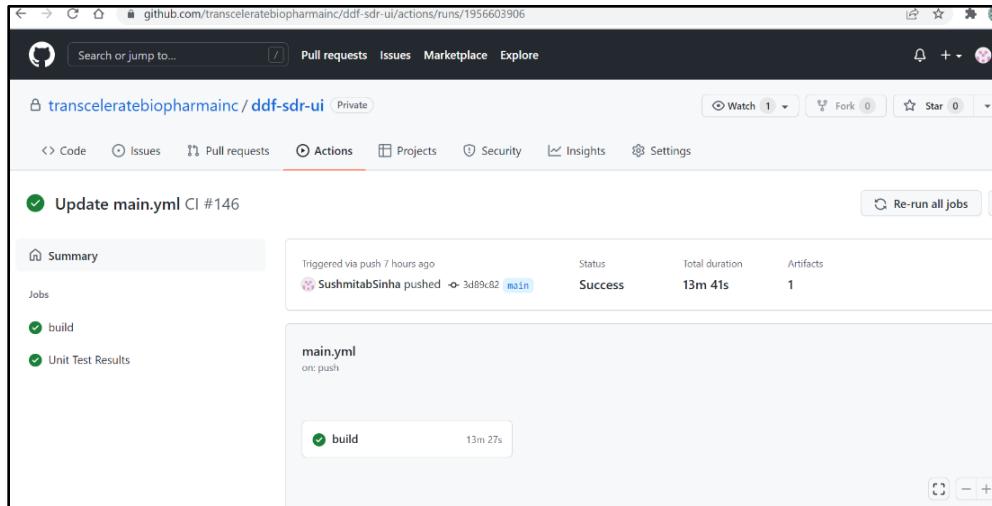
- Click on the workflow CI under All workflow.

Figure 43 GitHub Actions - CI Workflow



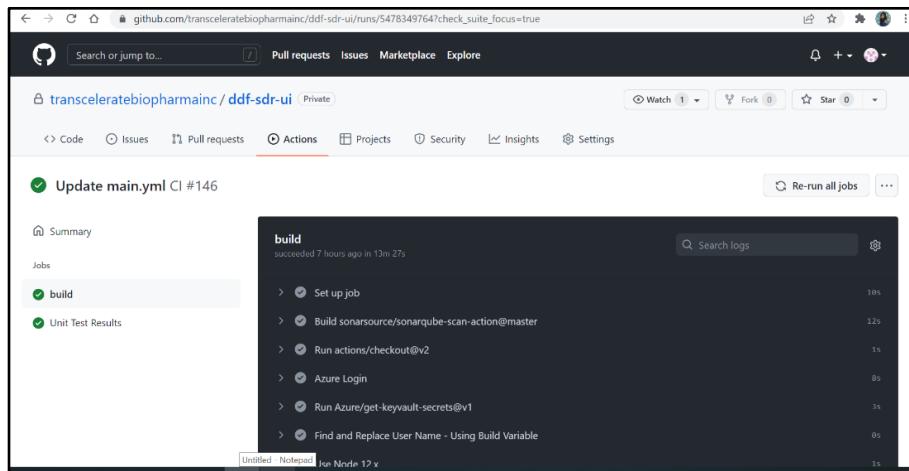
- Click on Run Workflow on the right-hand side. Then the action will be triggered.

Figure 44 GitHub - CI Workflow Run



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

Figure 45 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

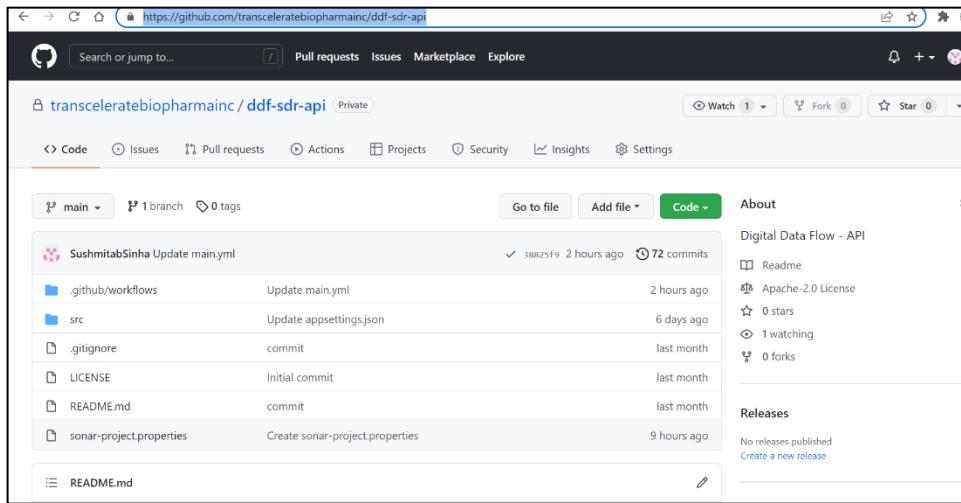
### PRE-REQUISITES

- Contributor access at Resource group Level.

### DEPLOYMENT STEPS:

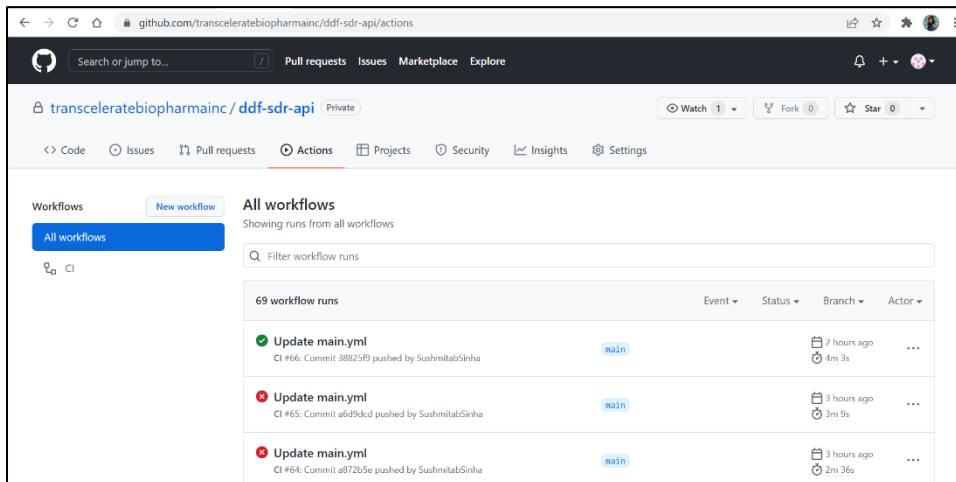
- Go to API repository on GitHub.

Figure 46 GitHub SDR API Repo



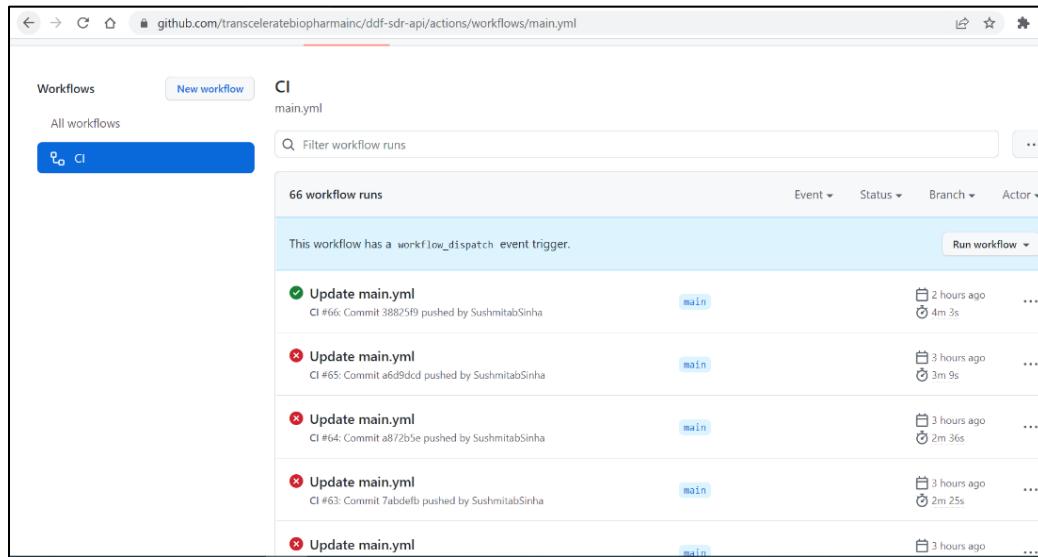
ii. Click on Actions tab.

Figure 47 GitHub Actions CI Workflow



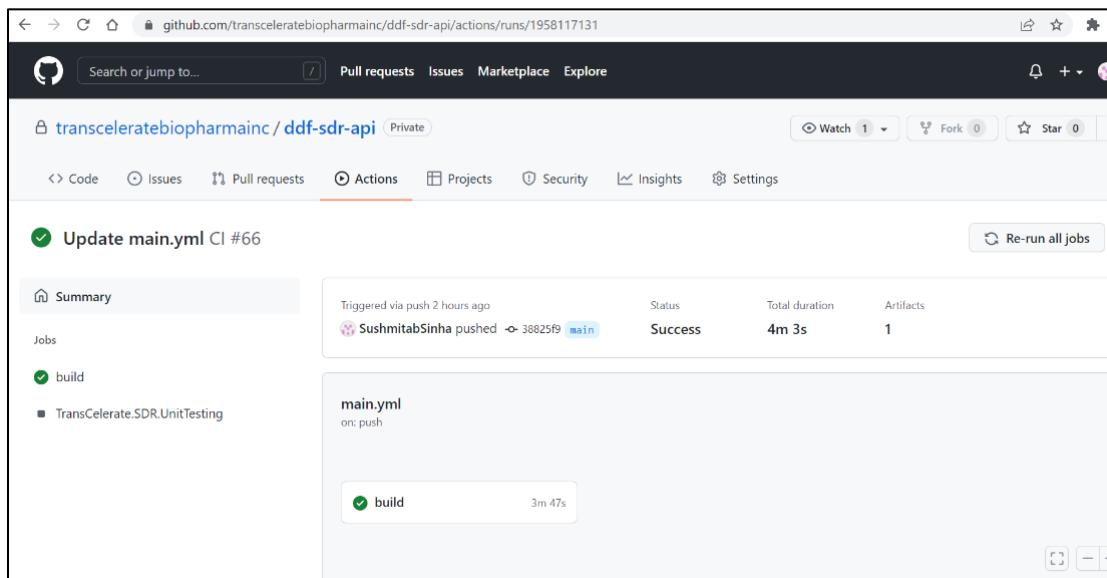
iii. Click on the workflow CI under All workflow.

Figure 48 GitHub CI Workflow Run



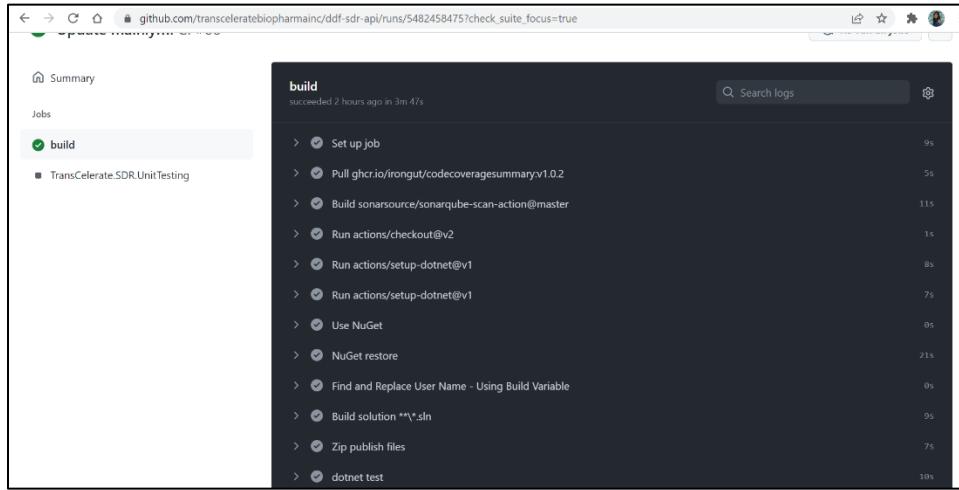
iv. Click on Run Workflow on the right-hand side. Then the action will be triggered.

Figure 49 GitHub CI Workflow Run



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

Figure 50 GitHub CI Workflow Output



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

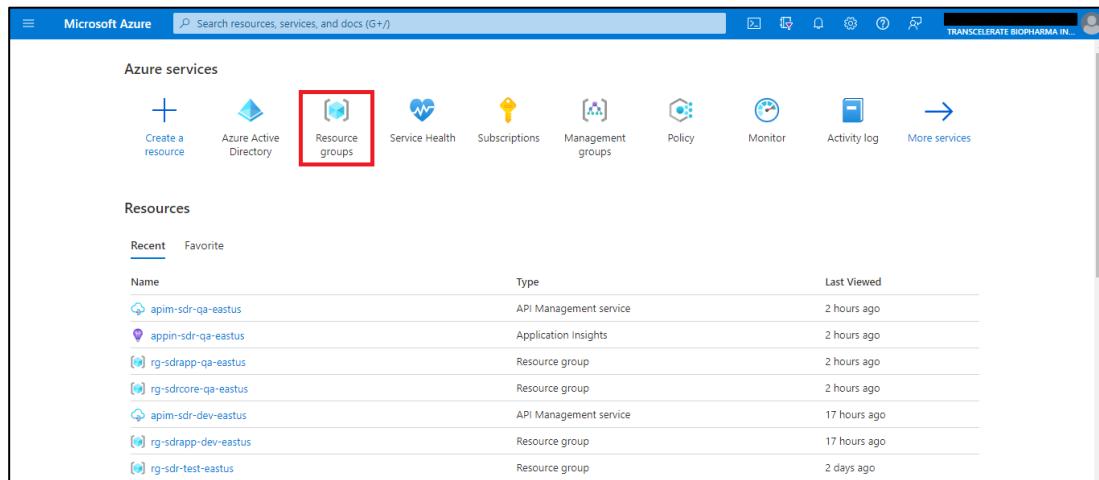
## 4.4 Deployment Verification

### UI APPLICATION VERIFICATION STEPS:

This is to verify the UI Application deployment was successful.

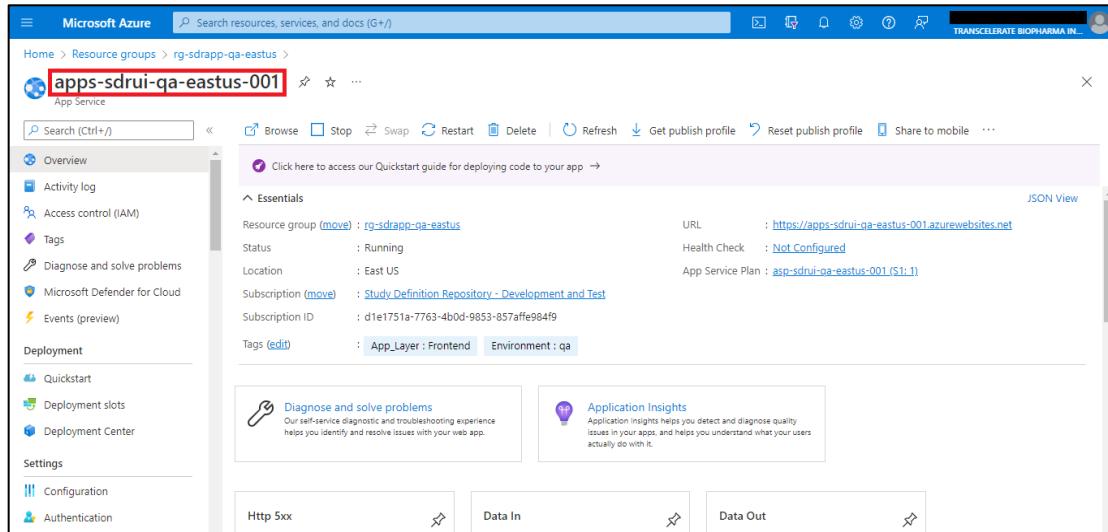
- Go to portal.azure.com. Click on resource group.

Figure 51 Azure Portal



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

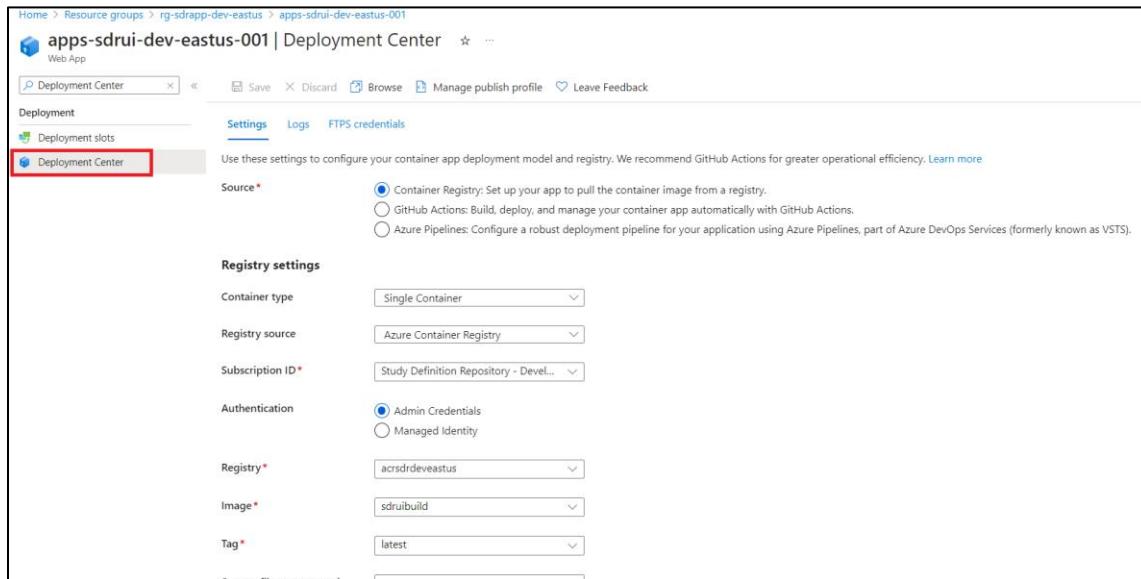
Figure 52 Azure Portal - SDR UI App Service



The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar with the text 'Search resources, services, and docs (G+/-)' and a user profile icon. Below the search bar, the URL is 'Home > Resource groups > rg-sdrapp-qa-eastus > apps-sdruui-qa-eastus-001'. The main content area is titled 'apps-sdruui-qa-eastus-001' and shows the 'App Service' blade. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Microsoft Defender for Cloud, Events (preview), Deployment (with sub-options Quickstart, Deployment slots, Deployment Center), Settings, Configuration, and Authentication. The 'Overview' section contains details such as Resource group (rg-sdrapp-qa-eastus), Status (Running), Location (East US), Subscription (Study Definition Repository - Development and Test), Subscription ID (d1e1751a-7763-4b0d-9853-857affe984f9), and Tags (App\_Layer : Frontend, Environment : qa). There are also links for 'Diagnose and solve problems' and 'Application Insights'.

iii. In search box, search for Deployment Center.

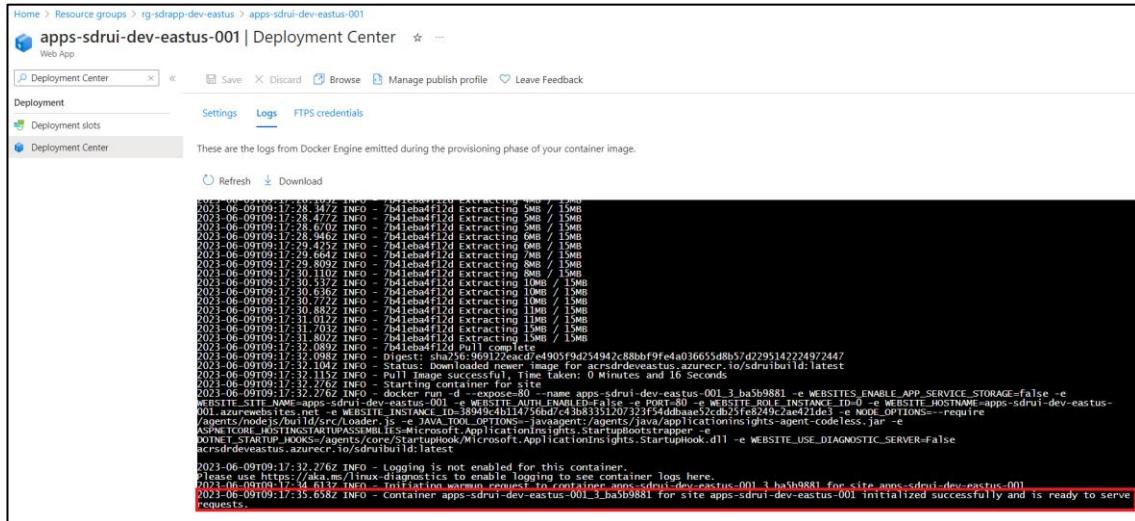
Figure 53 SDR UI App Service - Deployment Center Details



The screenshot shows the 'Deployment Center' blade for the app service 'apps-sdruui-dev-eastus-001'. The navigation bar includes tabs for Deployment Center, Deployment slots, and Deployment Center (which is selected and highlighted with a red box). Below the tabs, there are buttons for Save, Discard, Browse, Manage publish profile, and Leave Feedback. The main content area is titled 'Deployment Center' and contains sections for 'Source', 'Registry settings', and 'Logs'. Under 'Source', there are three options: Container Registry (selected), GitHub Actions, and Azure Pipelines. Under 'Registry settings', there are fields for Container type (Single Container), Registry source (Azure Container Registry), Subscription ID (Study Definition Repository - Dev...), Authentication (Admin Credentials selected), Registry (acrsdruideastus), Image (sdruibuild), and Tag (latest). A note at the bottom says 'Container file or command'.

iv. Click on Logs.

Figure 54 SDR UI App Service Deployment Center -Logs



```

2023-06-09T09:17:28.347Z INFO - 7b1leba4f12d Extracting 8MB / 15MB
2023-06-09T09:17:28.348Z INFO - 7b1leba4f12d Extracting 8MB / 15MB
2023-06-09T09:17:28.602Z INFO - 7b1leba4f12d Extracting 8MB / 15MB
2023-06-09T09:17:28.946Z INFO - 7b1leba4f12d Extracting 8MB / 15MB
2023-06-09T09:17:29.001Z INFO - 7b1leba4f12d Extracting 8MB / 15MB
2023-06-09T09:17:29.664Z INFO - 7b1leba4f12d Extracting 7MB / 15MB
2023-06-09T09:17:29.809Z INFO - 7b1leba4f12d Extracting 8MB / 15MB
2023-06-09T09:17:30.008Z INFO - 7b1leba4f12d Extracting 8MB / 15MB
2023-06-09T09:17:30.337Z INFO - 7b1leba4f12d Extracting 10MB / 15MB
2023-06-09T09:17:30.636Z INFO - 7b1leba4f12d Extracting 10MB / 15MB
2023-06-09T09:17:30.882Z INFO - 7b1leba4f12d Extracting 11MB / 15MB
2023-06-09T09:17:31.012Z INFO - 7b1leba4f12d Extracting 11MB / 15MB
2023-06-09T09:17:31.142Z INFO - 7b1leba4f12d Extracting 11MB / 15MB
2023-06-09T09:17:31.480Z INFO - 7b1leba4f12d Extracting 11MB / 15MB
2023-06-09T09:17:31.802Z INFO - 7b1leba4f12d Extracting 11MB / 15MB
2023-06-09T09:17:32.002Z INFO - 7b1leba4f12d Extracting 11MB / 15MB
2023-06-09T09:17:32.104Z INFO - Status: Downloaded newer image for acrstorage.azurecr.io/sdrui:build:latest
2023-06-09T09:17:32.115Z INFO - Pull Image successful. Time taken: 0 Minutes and 16 Seconds
2023-06-09T09:17:32.115Z INFO - Container image pulled successfully for container 'sdrui'.
2023-06-09T09:17:32.276Z INFO - docker run -d --expose=80 --name apps-sdrui-dev-eastus-001_3_bash9881 -e WEBSITES_ENABLE_APP_SERVICE_STORAGE=false -e APP_NAME=apps-sdrui-dev-eastus -e WEBSITE_HOSTNAME=apps-sdrui-dev-eastus-001.azurewebsites.net -e WEBSITE_SITE_NAME=apps-sdrui-dev-eastus -e WEBSITE_INSTANCE_ID=38949c4b14750bd434b3a313120732f44dbbaae52d721fe8249c2ae421dc3 -e NODE_OPTIONS=--require=/agents/nodejs/build/src/loader.js -e JAVA_TOOL_OPTIONS=-javaagent:/agents/java/applicationinsights-agent-codeless.jar -e APPINSIGHTS_INSTRUMENTATIONKEY=49059d54942c88bbf9fe4a03655d8b57d2295142224977447 -e APPINSIGHTS_DWORD_HOOKS=agents/core/Stampede/Microsoft.ApplicationInsights.StartupHook.DLL -e WEBSITE_USE_DIAGNOSTIC_SERVER=False
2023-06-09T09:17:32.276Z INFO - Logging is not enabled for this container.
Please use https://aka.ms/linux-diagnostics to enable logging to see container logs here.
2023-06-09T09:17:35.658Z INFO - Container apps-sdrui-dev-eastus-001_3_bash9881 for site apps-sdrui-dev-eastus-001 initialized successfully and is ready to serve requests.

```

## SDR API BACK-END APP AND FUNCTION APP VERIFICATION:

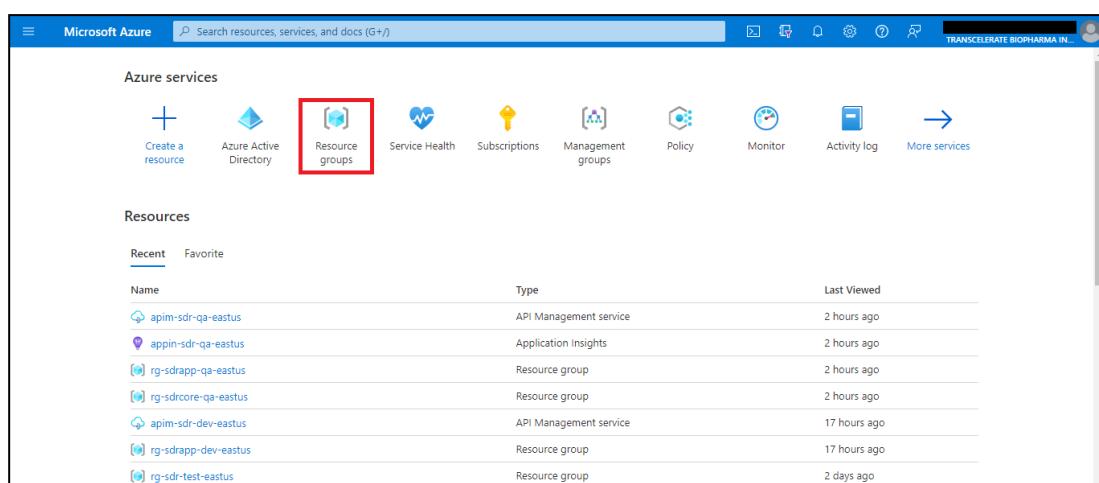
The same steps as mentioned above for SDR UI Application verification can be followed for SDR API deployment verification as well, in the corresponding App Service instance.

### FUNCTION APP VERIFICATION STEPS:

This is to verify the Function App deployment was successful.

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

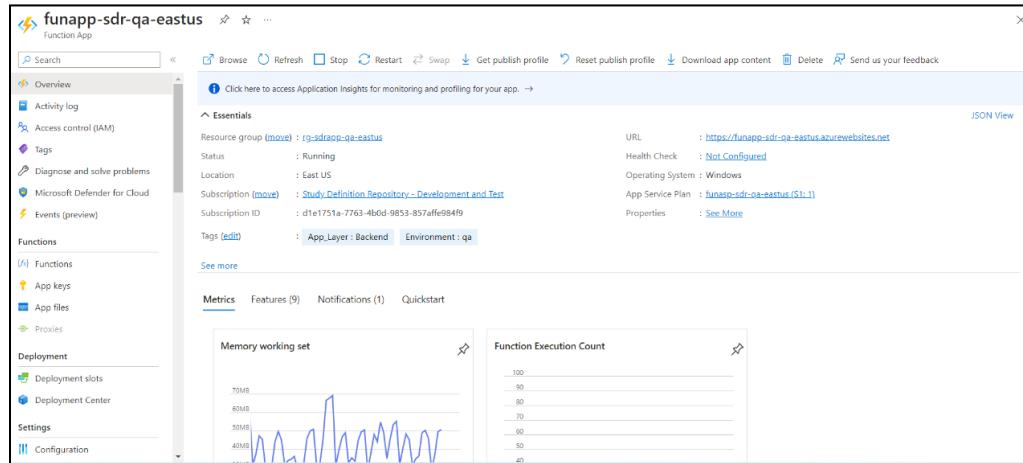
Figure 55 Azure Portal



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

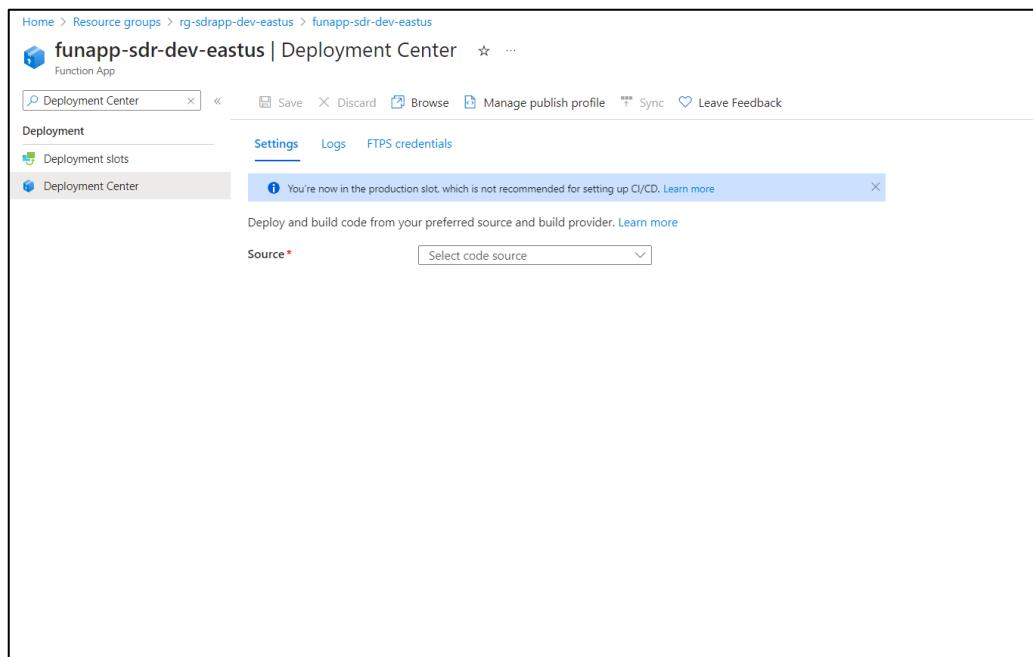
- ii. Select the required resource group and select the Function App Service instance.

*Figure 56 Azure Portal - SDR Function App Service*



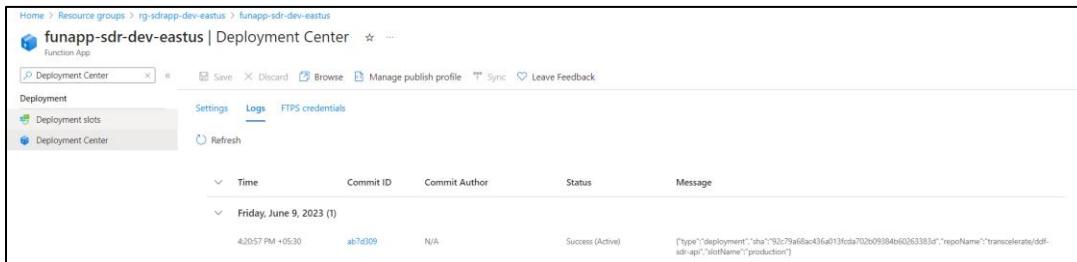
- iii. In search box, search for Deployment Center.

*Figure 57 SDR Function App Service – Deployment Center*



- iv. Click on Logs

Figure 58 SDR Function App Service Deployment Center – Logs



Time	Commit ID	Commit Author	Status	Message
Friday, June 9, 2023 (1)	ab7d309	N/A	Success (Active)	["type":"deployment","sha":"9c279af68ac036a013cd0702b09384b60263383d","repoName":"transcelerate/dsf-sdr-api","slotName":"production"]

## 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.

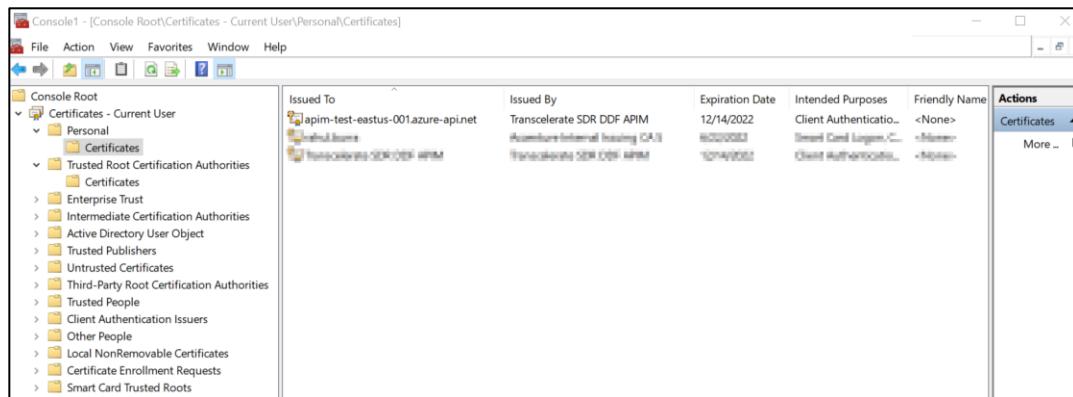
### CREATE CLIENT CERTIFICATE:

- Create self-signed certificate for authentication.

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

- 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 59 Client Certificate

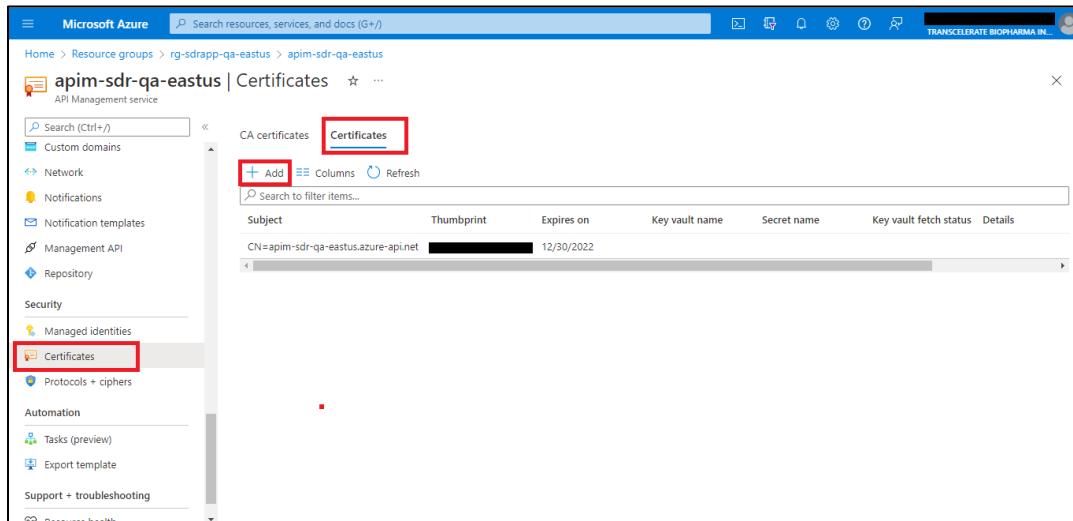


- Export the certificate in .pfx format and set password when prompted.

#### UPLOAD THE CLIENT CERTIFICATE TO APIM:

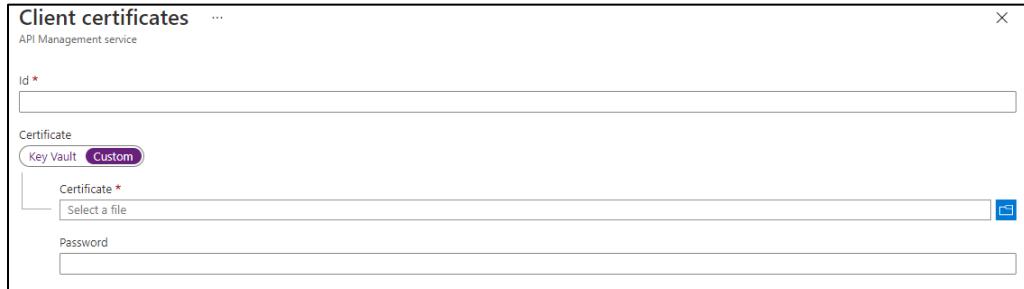
- In Azure Portal, Go to the Certificates option under the “Security” section of APIM. Go to “Certificates” option and click on “Add” option

Figure 60 APIM Certificates



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

Figure 61 APIM Certificates - Client Certificates

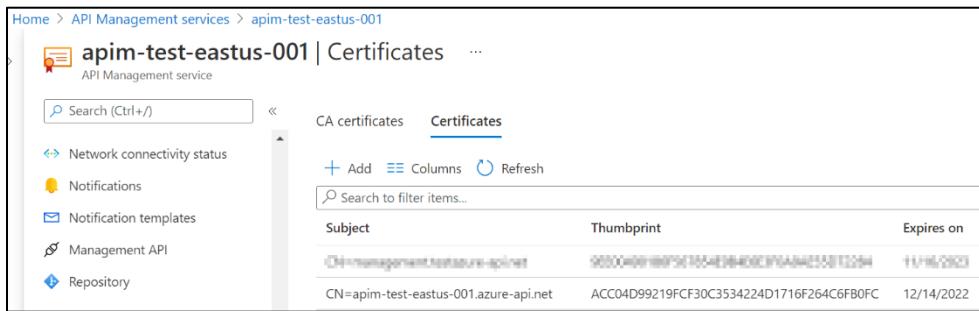


The screenshot shows the 'Client certificates' configuration page. It includes fields for 'Id \*' (mandatory), 'Certificate' (with 'Key Vault' and 'Custom' options), 'Select a file' for the certificate, and a 'Password' field.

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

### Client Certificate

Figure 62 APIM Certificates - Client Certificate



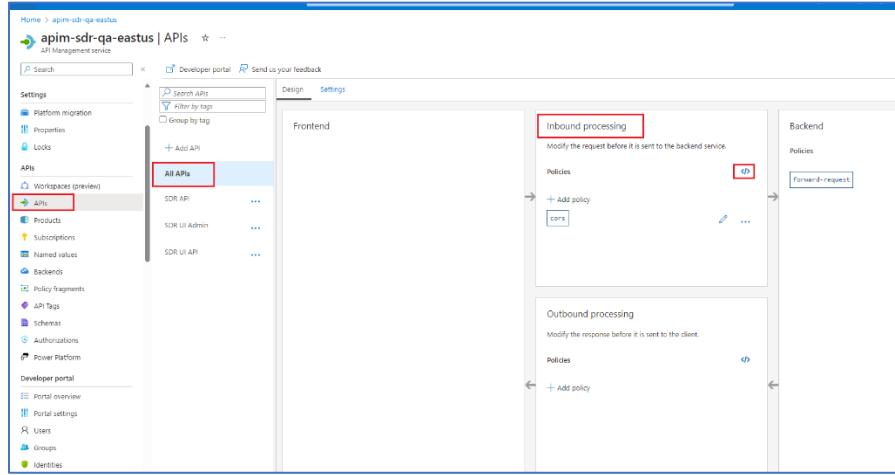
The screenshot shows the 'Certificates' blade in the API Management service 'apim-test-eastus-001'. It displays a table of certificates:

Subject	Thumbprint	Expires on
CN=management.azure-api.net	00000000000000000000000000000000	11/16/2023
CN=apim-test-eastus-001.azure-api.net	ACC04D99219FCF30C3534224D1716F264C6FB0FC	12/14/2022

### 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 63 : APIM Inbound Policy for All APIs



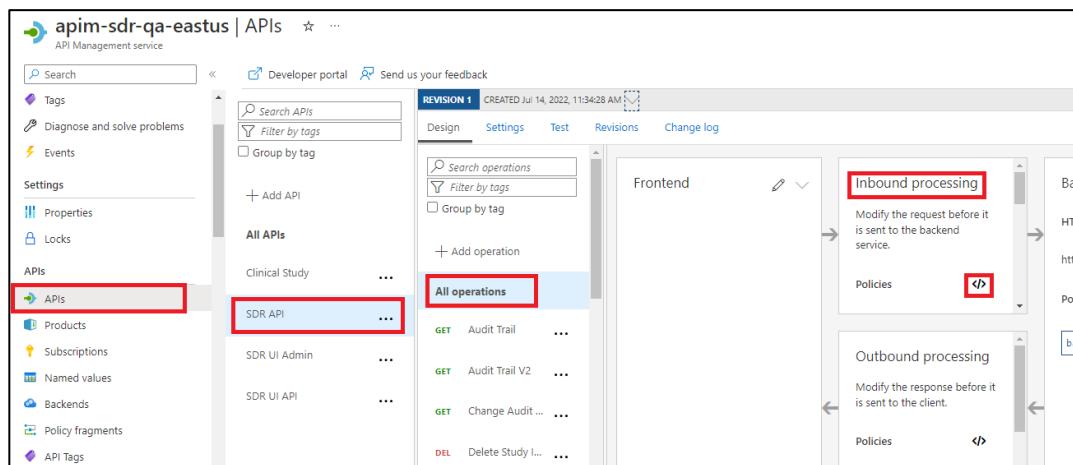
iii. Add the policy code below to validate the Incoming requests.

```
<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 64 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 65 : Update the Api-Key header name

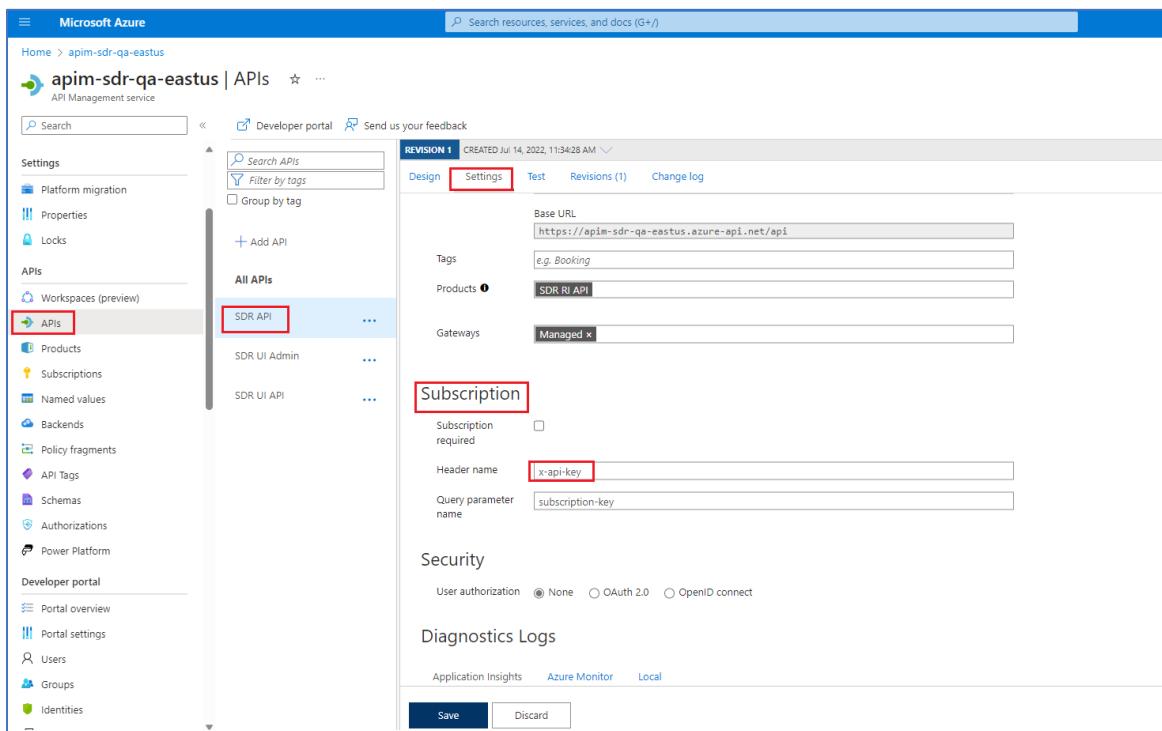
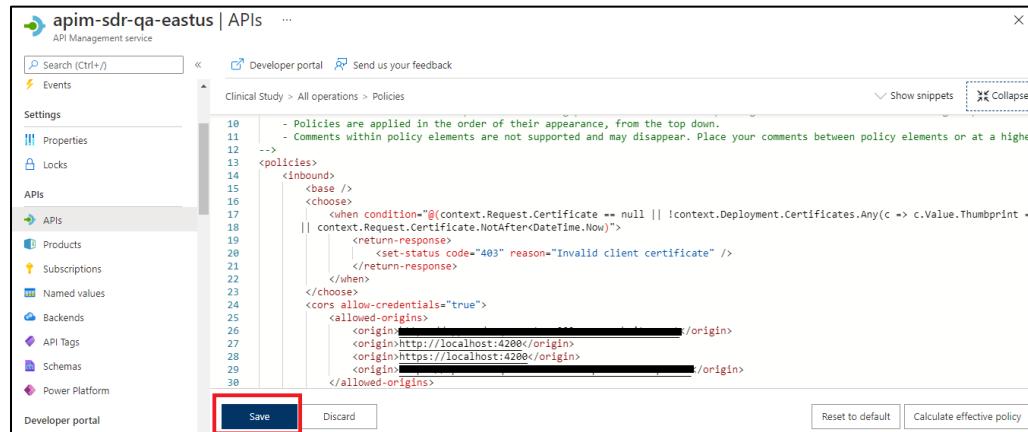


Figure 66 APIM - Inbound Policy



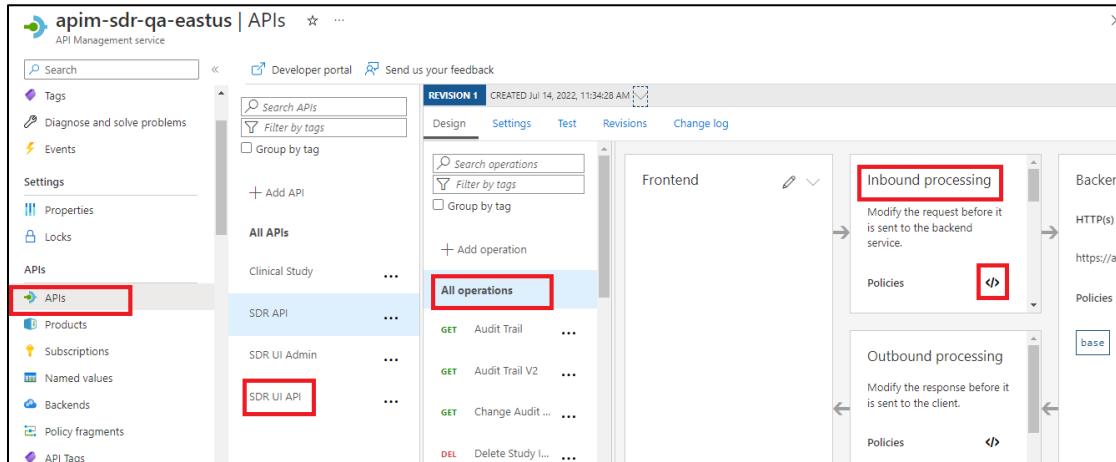
```

apim-sdr-qa-eastus | APIs ... < Search (Ctrl+ /) > Developer portal Send us your feedback
Clinical Study > All operations > Policies Show snippets Collapse
- Policies are applied in the order of their appearance, from the top down.
- Comments within policy elements are not supported and may disappear. Place your comments between policy elements or at a higher level.
<policies>
  <inbound>
    <base />
    <choose>
      <when condition="@{context.Request.Certificate == null || !context.DeploymentCertificates.Any(c => c.Value.Thumbprint == context.Request.Certificate.NotAfter < DateTime.Now)}">
        <set-status code="403" reason="Invalid client certificate" />
      </when>
    </choose>
    <cors allow-credentials="true">
      <allowed-origins>
        <origin>[REDACTED]/origin</origin>
        <origin>http://localhost:4200</origin>
        <origin>https://localhost:4200</origin>
        <origin>[REDACTED]</origin>
      </allowed-origins>
    </cors>
  </inbound>
</policies>
  
```

Developer portal Save Discard Reset to default Calculate effective 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";
    }" />
  </inbound>
</policies>
  
```

```

        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

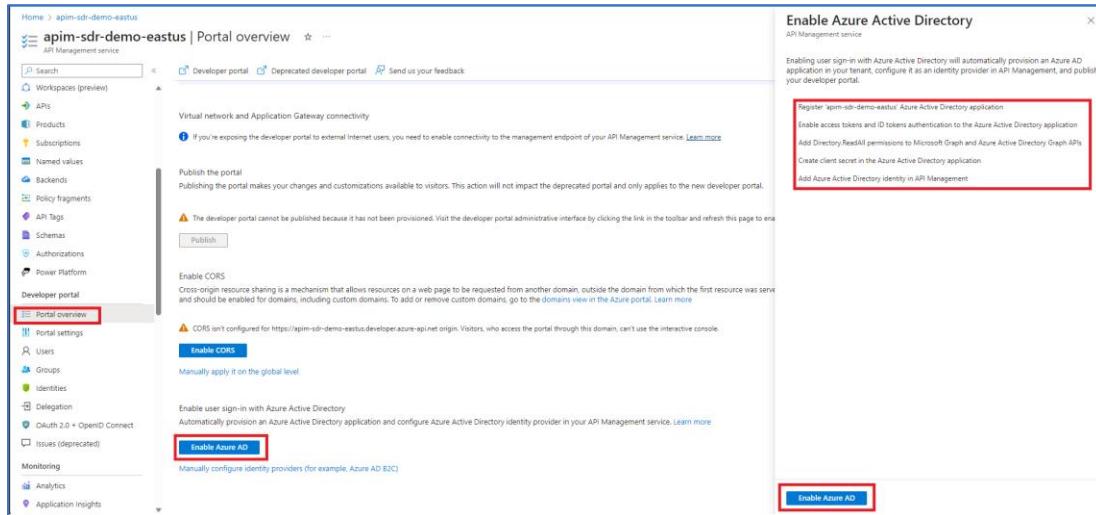
APIM Developer Portal can be used to generate API-Key for SDR API access. For accessing the developer portal and generating the API-Key, few additional APIM resource

configurations to be added as mentioned below apart from the steps automated through terraform

### 5.1.1. Add App registration for Azure AD login

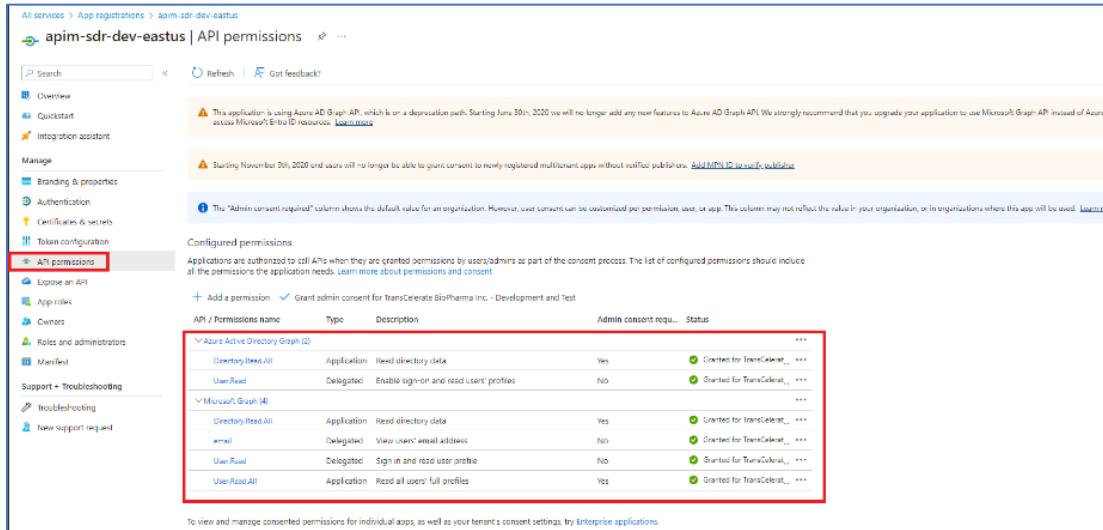
- i. In Azure Portal, goto APIM and select Identities.
- ii. Remove Azure Active Directory provider type.
- iii. Select Portal Overview.
- iv. Click on Enable Azure AD button.
- v. The Enable Azure Active Directory dialog will show the list of activites that will be done as a part of enabling the Azure AD.
- vi. Click Enable Azure AD button.

Figure 67 : Enable Azure AD for Developer Portal



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

Figure 68 : API Permissions for APIM App Registration

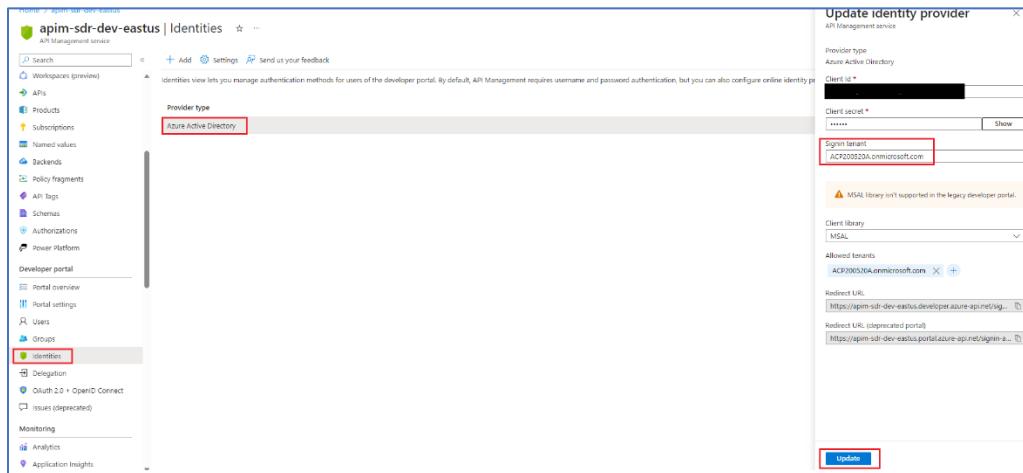


API / Permissions name	Type	Description	Admin consent req.	Status
Directory.Read.All	Application	Read directory data	Yes	Granted for TransCelerate...
User.Read	Delegated	Enable sign-on and read user profiles	No	Granted for TransCelerate...
Microsoft Graph (4)				
Directory.Read.All	Application	Read directory data	Yes	Granted for TransCelerate...
email	Delegated	View user's email address	No	Granted for TransCelerate...
User.Read	Delegated	Sign in and read user profile	No	Granted for TransCelerate...
User.Read.All	Application	Read all user's full profiles	Yes	Granted for TransCelerate...

### 5.1.2. Identities in APIM

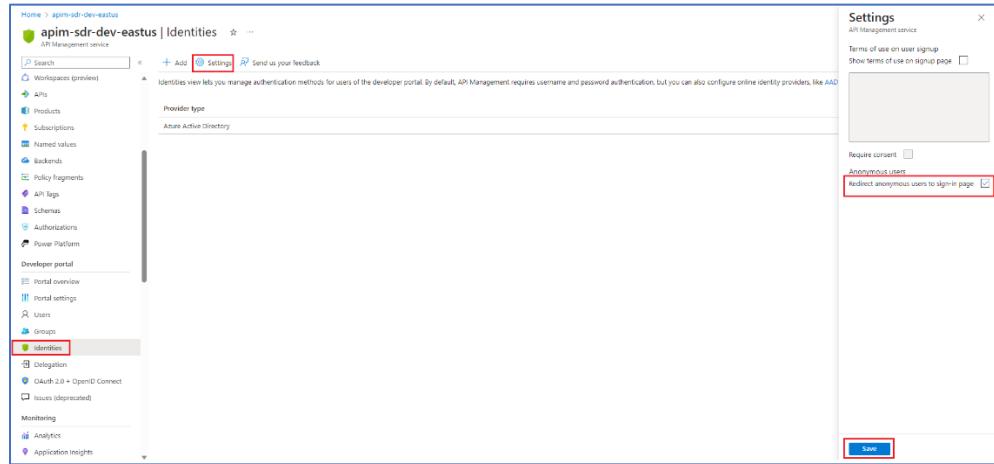
- i. In Azure Portal, goto APIM and select Identities.
- ii. Select Azure Active Directory.
- iii. Add the primary domain in the Signin tenant value.
- iv. Then click Update.

Figure 69 : 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.

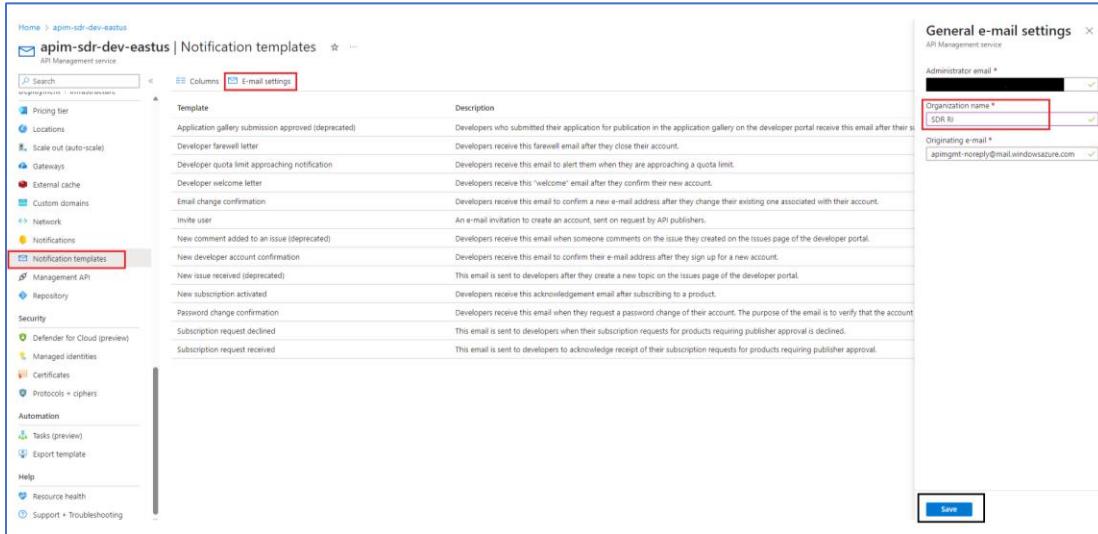
Figure 70 : Redirect anonymous users to sign-in page



### 5.1.3. Change Publisher Name in APIM

- In Azure Portal, goto APIM and select Notification Templates.
- Select E-mail Settings and update the Organization name to update the publisher name of the APIs.
- Then click save.

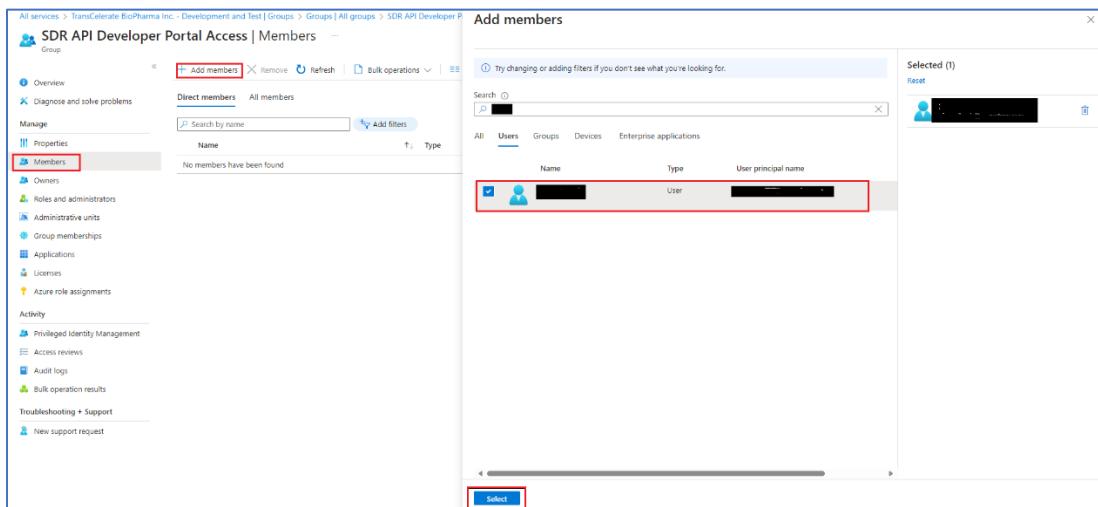
Figure 71 : Change Publisher Name



### 5.1.4. Add User to the AD Group for Developer Portal Access

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

Figure 72 : Add User to Azure AD Group



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