



Study Definition Repository (SDR) Reference Implementation

Azure Setup and Deployment Guide Version 2.0

Document History

Version No.	Date	Author	Revision Description
V1.0	15-Mar-2022	Infra Team	Initial Draft
V2.0	29-Jun-2022	Mani / Jabina	Extracted Smoke Testing section

Table of Contents

1. Introduction.....	6
1.1. Document Scope	6
1.2. Out of Scope.....	6
1.3. Audience.....	6
1.4. Pre-Requisites	6
1.5. Definitions and Acronyms	6
2. Azure Infrastructure.....	7
2.1. Resource Provider Registration	7
2.2. Create Azure AD Groups	10
2.3. Create Users.....	12
2.4. Role Based Access Controls (RBAC).....	13
2.5. Storage Account and Service Principal Configuration in Azure	16
2.6. Adding Secrets in GitHub.....	17
2.7. Execute GitHub Action for IaC deployment	20
2.8. Manual Configuration Changes on Azure Platform.....	20
2.8.1. Database	20
2.8.2. PaaS Setup – OAuth 2.0 Configuration for SDR UI Application:.....	23
2.8.3. UI App Path Mapping.....	29
2.8.4. API Management	30
2.8.5. Key Vault	31
3. Resource Validation	36
3.1. Low-Level Design Document	36
3.2. Virtual Network	37
3.3. Subnet	41
3.4. Delegated Subnet	42
3.5. Other Resources.....	44
4. Application Code Deployment	44
4.1 GitHub Secrets used in WorkFlow	44
4.2 Deploy the UI Application	45
4.3 Deploy Back-End API.....	48
4.4 Deployment Verification	51
5. PaaS Setup.....	54
5.1. PaaS Setup for APIM.....	54
5.2. Secure APIs using client certificate authentication in API Management	58

List of Figures

Figure 1 Select Subscriptions in Azure Portal	8
Figure 2 Select Subscription	8
Figure 3 Resource Providers in a Subscription.....	9
Figure 4 Registering Resource Providers.....	9
Figure 5 Adding new groups	12
Figure 6 Add New Group	12
Figure 7 Create New User	13
Figure 8 Access Control (IAM)	14
Figure 9 Add Role Assignment.....	14
Figure 10 Select Roles.....	15
Figure 11 Select Members for Role Assignment	15
Figure 12 View Role Assignments.....	16
Figure 13 GitHub Actions Secrets	19
Figure 14 Add new GitHub Action Secret.....	20
Figure 15 CosmosDB Firewall and Virtual Networks	21
Figure 16 CosmosDB Data Explorer	22
Figure 17 Creating Collection in CosmosDB	22
Figure 18 CosmosDB Indexing	23
Figure 19 Azure AD - App Registration	24
Figure 20 Azure AD - Register Application.....	24
Figure 21 Azure AD - App Registration - Redirect URI.....	25
Figure 22 Azure AD - App Registration - Authentication.....	25
Figure 23 Azure AD - App Registration - Expose an API.....	26
Figure 24 Azure AD - App Registration - API Permission	26
Figure 25 Azure AD - App Registration - Add API Permission.....	27
Figure 26 Azure AD - App Registration - API Permission - Grant Admin Consent.....	27
Figure 27 Azure AD - App Registration - Certificates & Secrets	28
Figure 28 Azure AD - App Registration - Token Configuration	28
Figure 29 Azure AD - App Registration - Essential Secrets	29
Figure 30 Azure Portal - SDR App Resource Group	30
Figure 31 API Management - Add API	30
Figure 32 API Management - Create an HTTP API.....	31
Figure 33 Add Key Vault Access Policy.....	32
Figure 34 Select Secret Permissions in Access Policy in Key Vault	33
Figure 35 Select Principal (List of users) In Key Vault Access Policy	34
Figure 36 Create Secrets in Key Vault	35
Figure 37 Add Secrets values in Key Vault	35
Figure 38 Resource Naming Convention.....	36
Figure 39 SDR Resource Naming Convention	37
Figure 40 VNet Configurations	38
Figure 41 Virtual Network.....	39
Figure 42 Virtual Network - DDoS protection.....	39
Figure 43 Virtual Network - Tags.....	40
Figure 44 Virtual Network - Diagnostic Setting	40

Figure 45 Subnet Details.....	41
Figure 46 Subnet Details.....	42
Figure 47 Delegated Subnet-001 Details.....	43
Figure 48 Delegated Subnet-001 Service Endpoints Details	43
Figure 49 Delegated Subnet-002 Details.....	43
Figure 50 Delegated Subnet-002 Service Endpoints Details	44
Figure 35 TransCelerate GitHub Project	45
Figure 36 TransCelerate GitHub SDR UI Repo	46
Figure 37 SDR UI Repo - GitHub Actions.....	46
Figure 38 GitHub Actions - CI Workflow	47
Figure 39 GitHub - CI Workflow Run	47
Figure 40 GitHub CI Workflow Output	48
Figure 41 TransCelerate GitHub Project	48
Figure 42 TransCelerate GitHub SDR API Repo.....	49
Figure 43 GitHub Actions CI Worklflow	49
Figure 44 GitHub CI Workflow Run	50
Figure 45 GitHub CI Workflow Run	50
Figure 46 GitHub CI Workflow Output	51
Figure 47 Azure Portal	51
Figure 48 Azure Portal - SDR UI App Service	52
Figure 49 SDR UI App Service - Advanced Tools	52
Figure 50 SDR UI App Service Advanced Tools - Go.....	53
Figure 51 Azure Kudu Tool	53
Figure 52 SDR UI Deployed Files	54
Figure 67 API Management - Name HTTP API	55
Figure 68 API Management - HTTP API Operation.....	55
Figure 69 API Management - API Operation Details	56
Figure 70 API Management – API Operation - Add Parameter	56
Figure 71 API Management – API Operation - Add Query Parameter.....	57
Figure 72 API Management – API Operation - Add Headers.....	57
Figure 73 API Management - Custom Domains	59
Figure 74 API Management - Custom Domains - Gateway	59
Figure 75 Root CA Certificate in MMC	60
Figure 76 Client Certificate.....	61
Figure 77 APIM Certificates	62
Figure 78 APIM Certificates - Add CA Certificate	63
Figure 79 APIM Certificates - Client Certificates.....	63
Figure 80 APIM Certificates - Root CA.....	63
Figure 81 APIM Certificates - Client Certificate	64
Figure 82 APIM Inbound Processing.....	64
Figure 83 APIM - Inbound Policy.....	66

1. Introduction

This document details the steps required to set up a new environment for the Study Definition Repository – Reference Implementation (MVP release) on Microsoft Azure Cloud Platform. It provides details for Infrastructure setup, environment validation, deploying the Web Application for UI and API along with Smoke testing.

This document is organized sequentially including Infrastructure, PaaS, UI and API components setup and is presented in a dependency-based order. All the sections listed here are mandatory for the environment setup.

1.1. Document Scope

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

1.2. Out of Scope

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

1.3. 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.

1.4. Pre-Requisites

- Access to the low-level design document and basic understanding on how to use it.
- Access to the azure portal with required permissions (detailed in upcoming sections).
- Azure CLI Refer to [Install CLI](#)
- 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](#)

1.5. Definitions and Acronyms

Term / Abbreviation	Definition
AAD	Azure Active Directory
AD	Active Directory

API	Application Programming Interface
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
MS	Microsoft
PaaS	Platform-as-a-Service
RBAC	Role Based Access Control
REST	Representational State Transfer
SDR	Study Definition Repository
UI	User Interface
URL	Uniform Resource Locator
VNet	Virtual Network

2. Azure Infrastructure

2.1. Resource Provider Registration

GOAL:

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

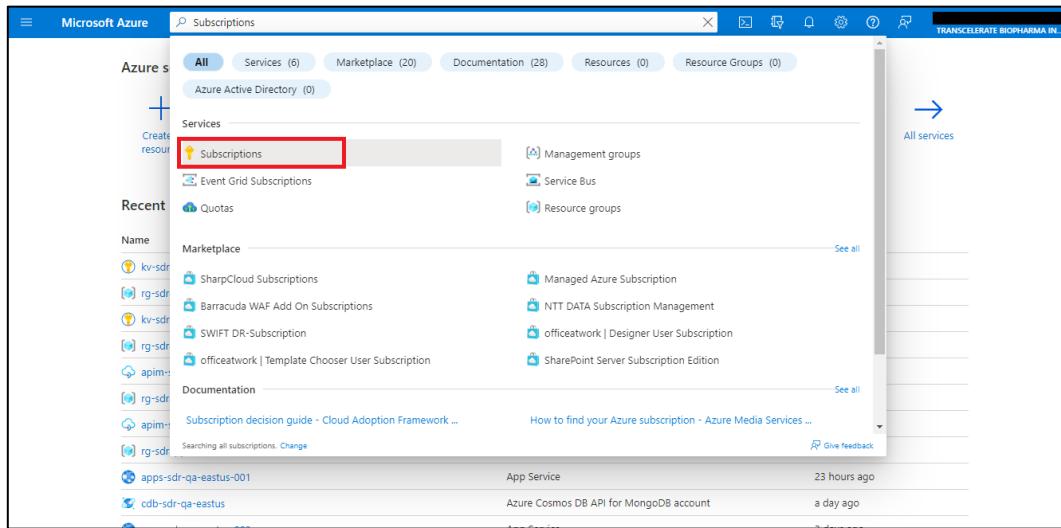
PRE-REQUISITES:

Global Admin or Subscription Owner level of access to Azure.

STEPS:

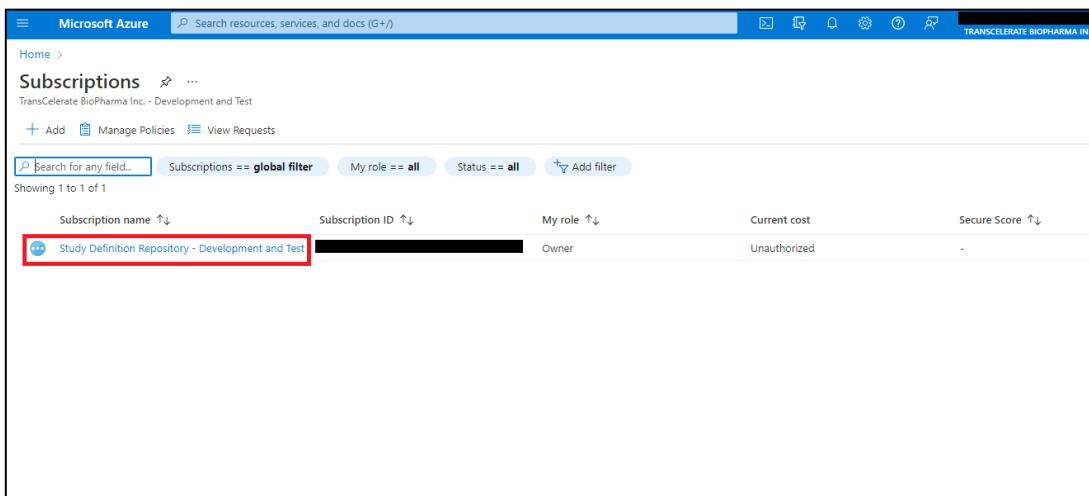
- i. Sign into the Azure portal.
- ii. On the Azure portal menu, search for Subscriptions. Select it from the available options as shown below.

Figure 1 Select Subscriptions in Azure Portal



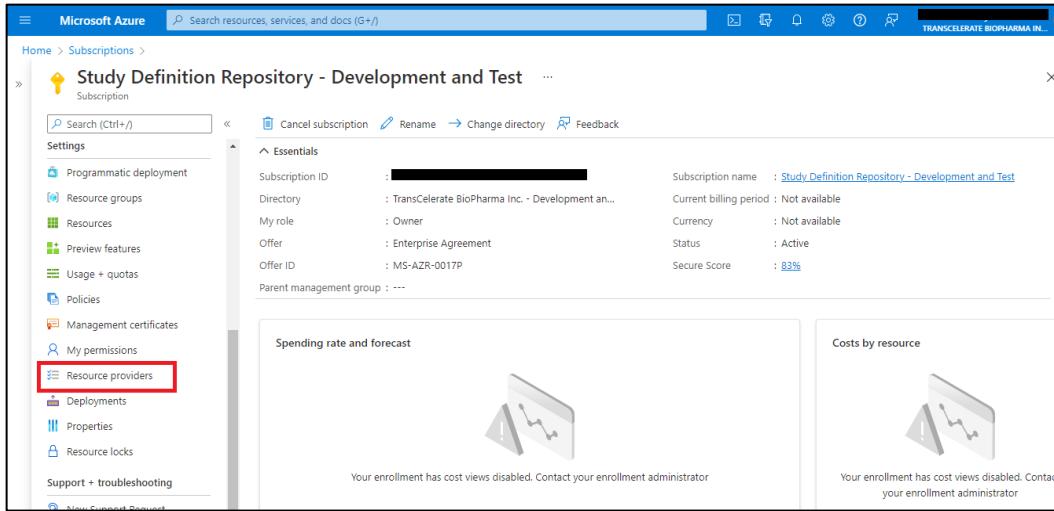
- iii. Select the subscription you want to view.

Figure 2 Select Subscription



- iv. On the left menu, under Settings, select Resource providers.

Figure 3 Resource Providers in a Subscription



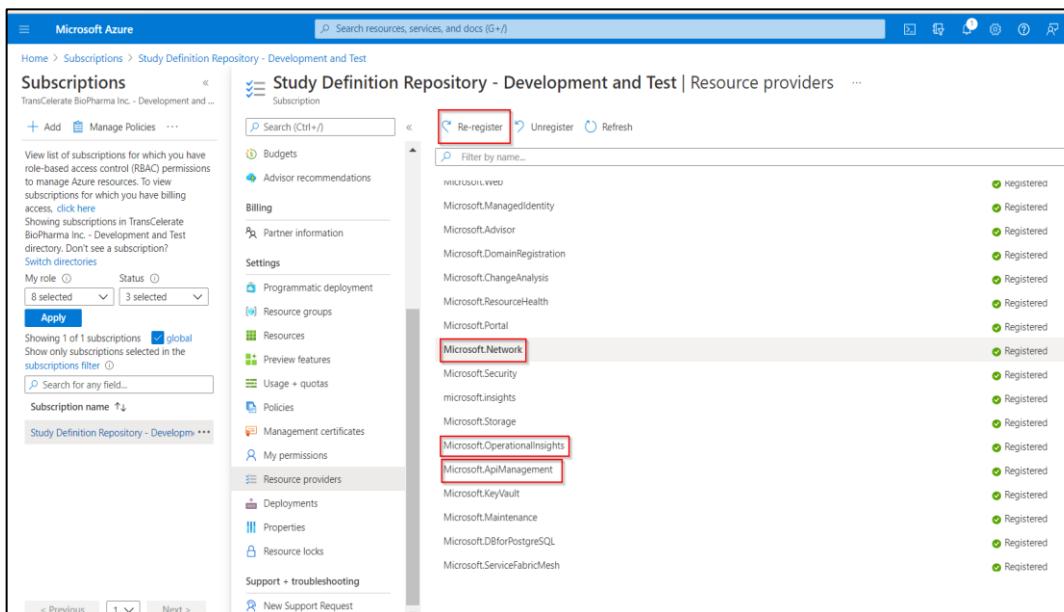
The screenshot shows the Microsoft Azure Subscriptions page for the 'Study Definition Repository - Development and Test' subscription. The 'Resource providers' section is highlighted with a red box. The page displays various subscription details like 'Subscription ID', 'Directory', 'My role', 'Offer', 'Offer ID', 'Parent management group', and 'Secure Score'. It also includes sections for 'Spending rate and forecast' and 'Costs by resource'.

- 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



The screenshot shows the 'Resource providers' list for the same subscription. The 'Microsoft.Network', 'Microsoft.OperationalInsights', and 'Microsoft.ApiManagement' entries are highlighted with red boxes. The 'Re-register' button is visible above the list.

Provider	Status
Microsoft.Network	Registered
Microsoft.OperationalInsights	Registered
Microsoft.ApiManagement	Registered
Microsoft.KeyVault	Registered
Microsoft.Maintenance	Registered
Microsoft.DBforPostgreSQL	Registered
Microsoft.ServiceFabricMesh	Registered

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.
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	
	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.

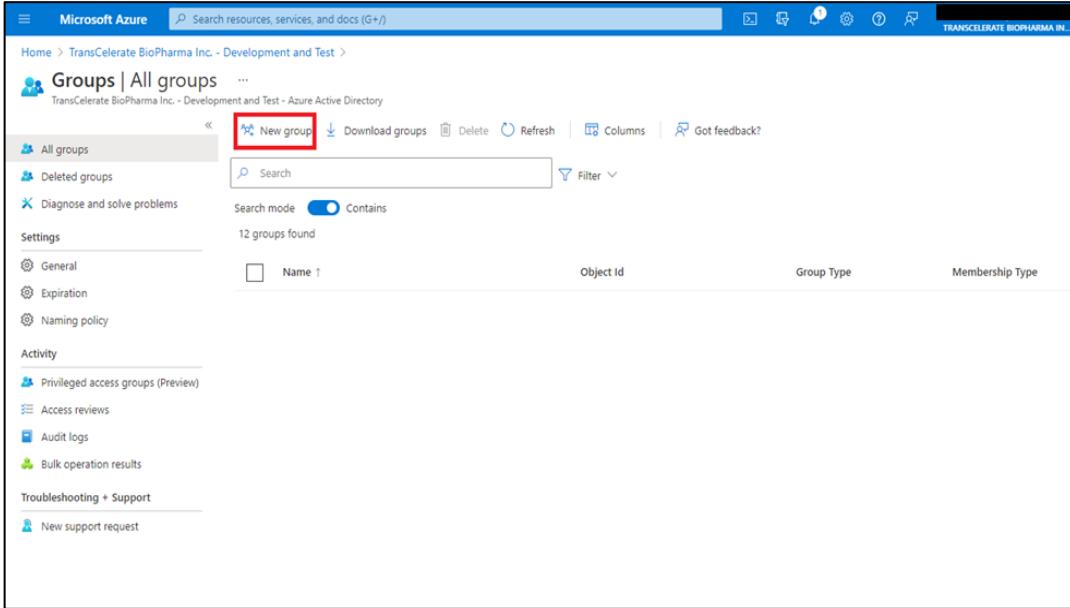
Note: To assign Azure AD roles to groups required Azure AD Premium P1 or P2 license, since this solution is part of MVP leveraged Azure AD Free plan. Follow the Microsoft Doc to assign Azure AD role to groups.

STEPS:

- i. Login to Azure portal
- ii. Search for Azure Active Directory
- iii. Click on the Groups on the left panel in AAD.

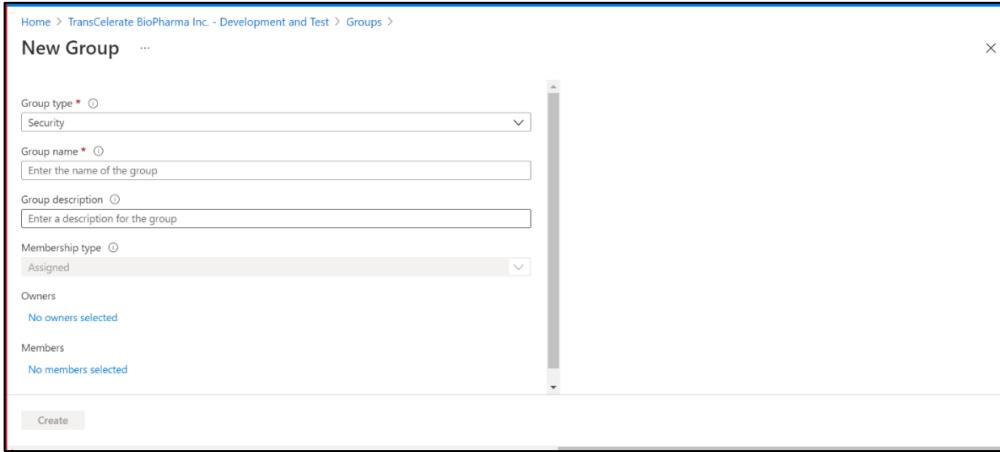
- iv. Click on New group tab on the top as shown below, add the security group and save the changes by adding the members to the group.

Figure 5 Adding new groups



The screenshot shows the Microsoft Azure portal interface for managing groups. The title bar says "Microsoft Azure". The main navigation bar includes "Search resources, services, and docs (G+)" and "TRANSCELERATE BIOPHARMA INC.". Below the search bar, the breadcrumb navigation shows "Home > TransCelerate BioPharma Inc. - Development and Test > Groups". The left sidebar has sections for "All groups", "Deleted groups", "Diagnose and solve problems", "Settings" (with options for General, Expiration, and Naming policy), "Activity" (with links for Privileged access groups (Preview), Access reviews, Audit logs, and Bulk operation results), and "Troubleshooting + Support" (with a link for New support request). The main content area is titled "Groups | All groups" and shows a table with columns: Name, Object Id, Group Type, and Membership Type. A red box highlights the "New group" button at the top of the table header. Below the table, there is a search bar and a "Filter" dropdown, and a note that 12 groups were found.

Figure 6 Add New Group



The screenshot shows the "New Group" dialog box. At the top, it says "Home > TransCelerate BioPharma Inc. - Development and Test > Groups > New Group". The dialog has several input fields: "Group type" (radio button selected for "Security"), "Group name" (text input field with placeholder "Enter the name of the group"), "Group description" (text input field with placeholder "Enter a description for the group"), "Membership type" (radio button selected for "Assigned"), and "Owners" (text input field showing "No owners selected"). Below these, there is a "Members" section with the text "No members selected". At the bottom right of the dialog is a "Create" button.

2.3. Create Users

GOAL:

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

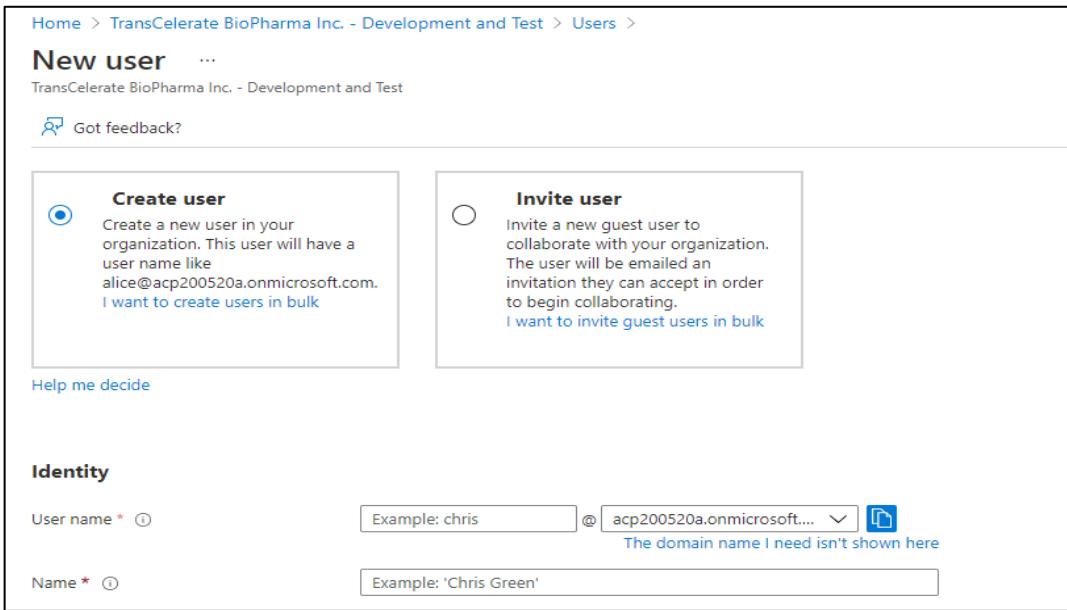
PRE-REQUISITES:

- 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



The screenshot shows the 'New user' creation interface in the Azure portal. At the top, there's a breadcrumb navigation: Home > TransCelerate BioPharma Inc. - Development and Test > Users >. Below it, the title 'New user' is displayed with a three-dot ellipsis. Underneath, it says 'TransCelerate BioPharma Inc. - Development and Test'. There's a 'Got feedback?' link with a feedback icon. Two large buttons are shown: 'Create user' (selected, indicated by a blue circle) and 'Invite user' (indicated by a grey circle). The 'Create user' button has the following text: 'Create a new user in your organization. This user will have a user name like alice@acp200520a.onmicrosoft.com. I want to create users in bulk'. The 'Invite user' button has the following text: 'Invite a new guest user to collaborate with your organization. The user will be emailed an invitation they can accept in order to begin collaborating. I want to invite guest users in bulk'. Below these buttons is a 'Help me decide' link. The 'Identity' section follows, with 'User name' and 'Name' fields. The 'User name' field contains 'chris' with an example of 'acp200520a.onmicrosoft....' and a dropdown arrow. The 'Name' field contains 'Chris Green' with an example of 'Chris Green'.

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:

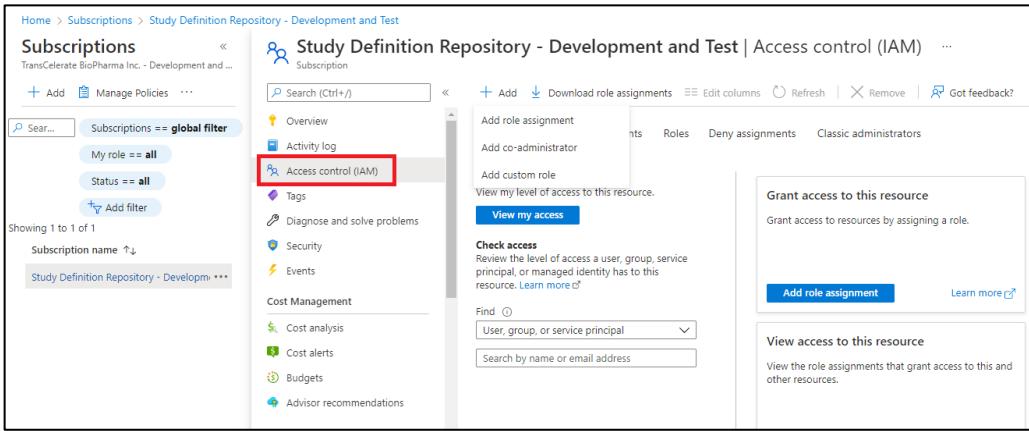
Access Control (IAM) is to limit access and assign roles to groups at the subscription, resource group, and resource level.

At subscription level

- i. Go to the subscription.

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

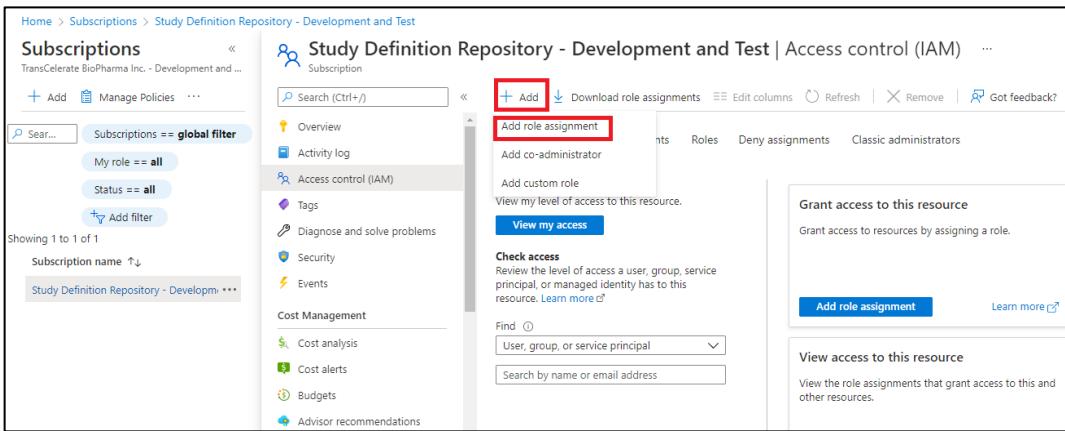
Figure 8 Access Control (IAM)



The screenshot shows the Azure Subscriptions page for 'Study Definition Repository - Development and Test'. The 'Access control (IAM)' option is highlighted with a red box. The main content area displays options like 'Add role assignment', 'Check access', and 'View access to this resource'.

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

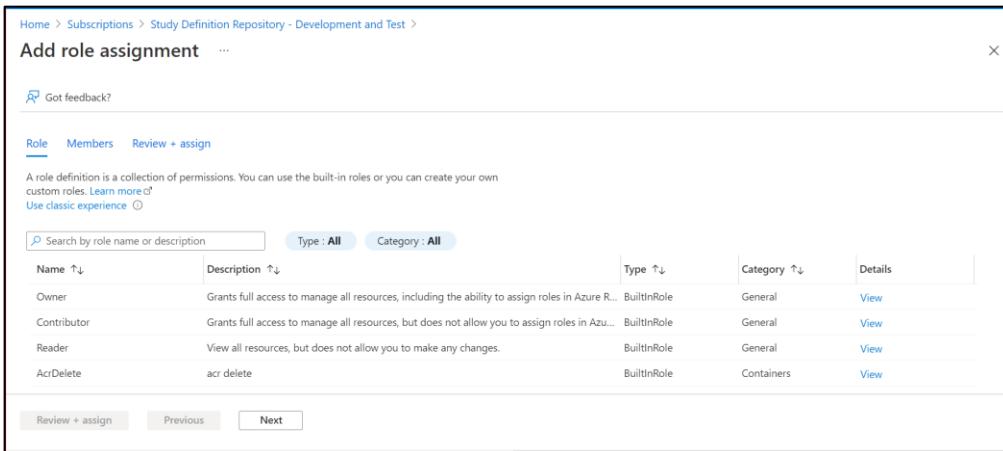
Figure 9 Add Role Assignment



The screenshot shows the same Azure Subscriptions page as Figure 8. The '+ Add' button for 'Add role assignment' is highlighted with a red box.

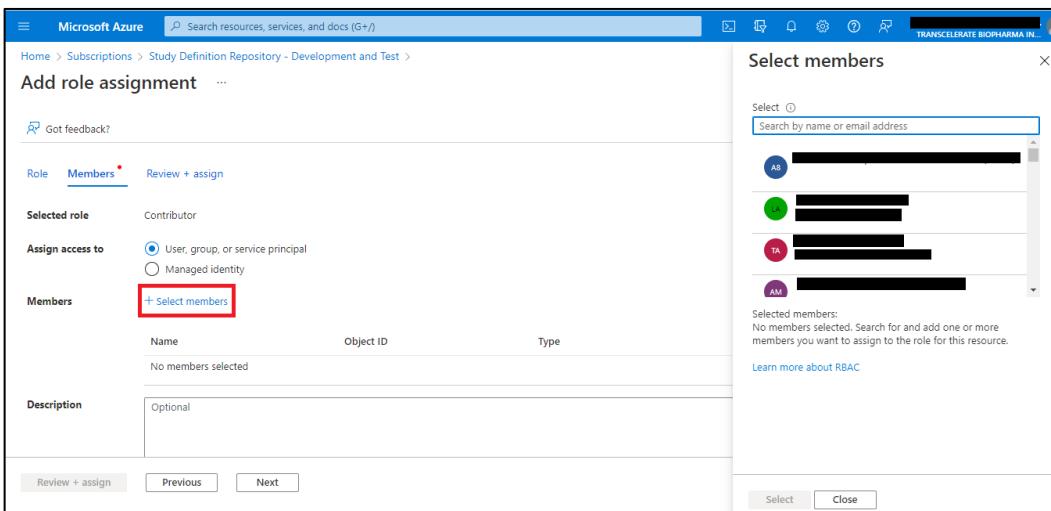
- iv. Select the required role (ex: reader, contributor etc.,) for the members and assign the role to the created groups as shown in the screenshot below and save the changes.

Figure 10 Select Roles



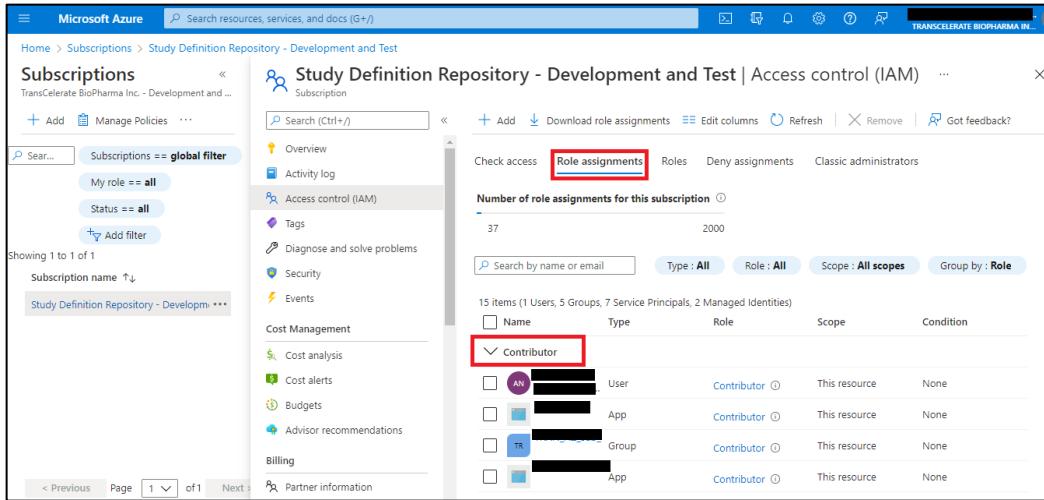
Name	Description	Type	Category	Details
Owner	Grants full access to manage all resources, including the ability to assign roles in Azure R...	BuiltinRole	General	View
Contributor	Grants full access to manage all resources, but does not allow you to assign roles in Azu...	BuiltinRole	General	View
Reader	View all resources, but does not allow you to make any changes.	BuiltinRole	General	View
acrDelete	acr 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 mentioned below:

Figure 12 View Role Assignments



Name	Type	Role	Scope	Condition
[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

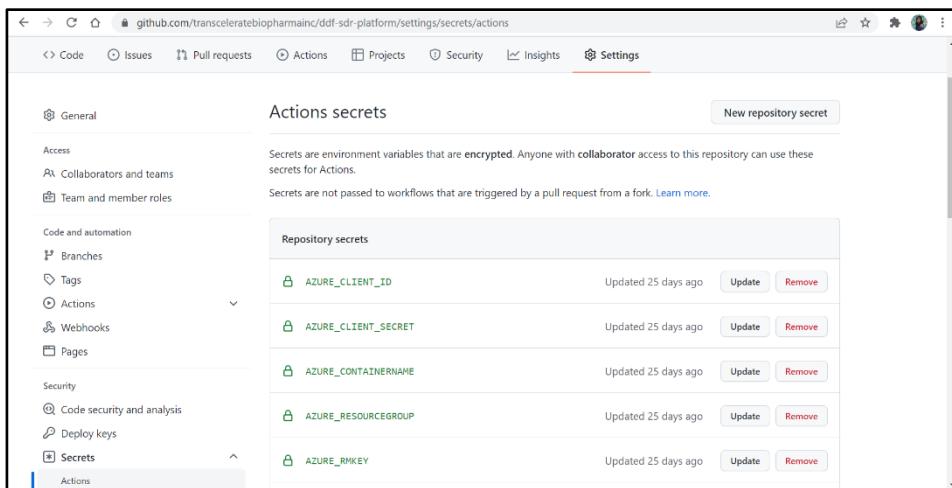
Secret Name	Values
AZURE_SP	The Service Principal details in JSON format.
AZURE_RESOURCEGROUP	The name of the Resource Group that contains the storage account.
AZURE_STORAGEACCOUNT	The Storage Account name
AZURE_CONTAINERNAME	The name of the blob container wherein the Terraform State file will be stored.
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
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).
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.

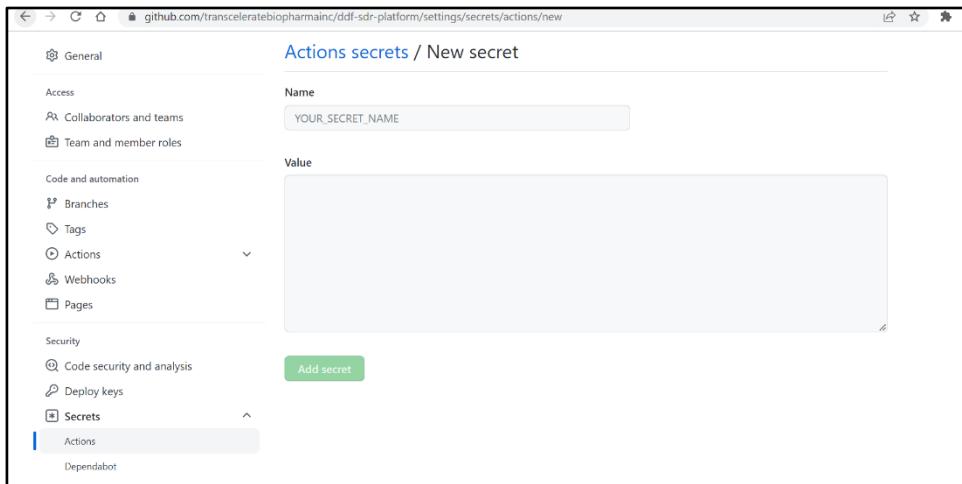
- i. Go to repo settings.
- ii. On the lower left-hand side of the screen, click on Secrets.
- iii. Under that click on Actions.
- iv. Then click on New Repository Secret.

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.

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. Database

GOAL:

- Update Cosmos DB firewall setting to enable access from Azure portal and Client machines.
- Create SDR Application database and collection(s) in Cosmos DB through Azure portal.

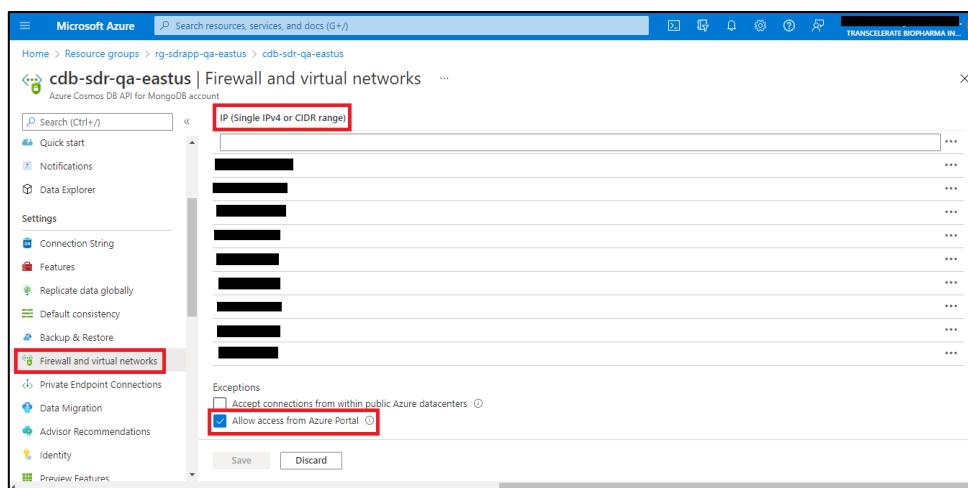
PRE-REQUISITES:

- Contributor level of access for Resource Group.

STEPS:

- i. In Cosmos DB settings under Firewall and virtual networks, add individual IP addresses for accessing the cosmos DB from client machines. (Refer section 8.3 for more details on adding individual IP addresses)
- ii. Check the option “Allow Access from Azure Portal” and click Save to save changes.

Figure 15 CosmosDB Firewall and Virtual Networks



- iii. Navigate to Cosmos DB – Data Explorer and add a collection with name “Study” under a new database with name “SDR” with settings mentioned below.

Figure 16 CosmosDB Data Explorer

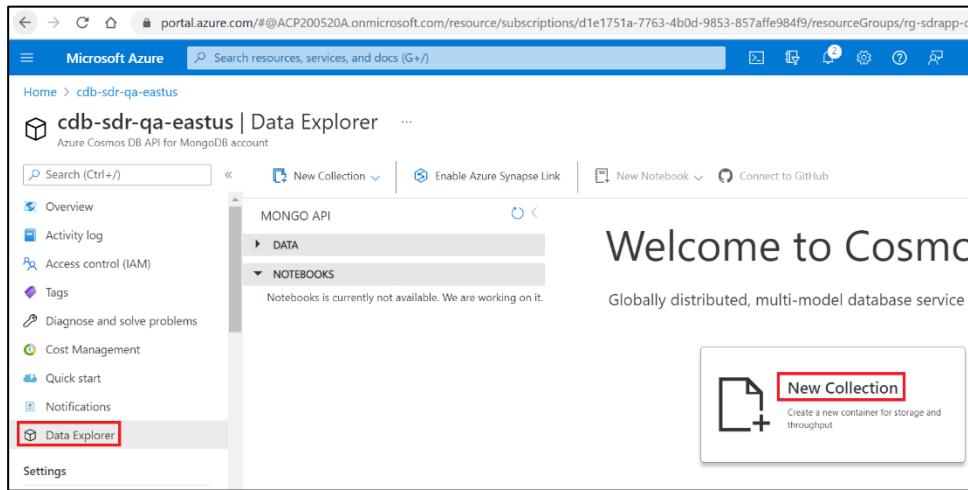
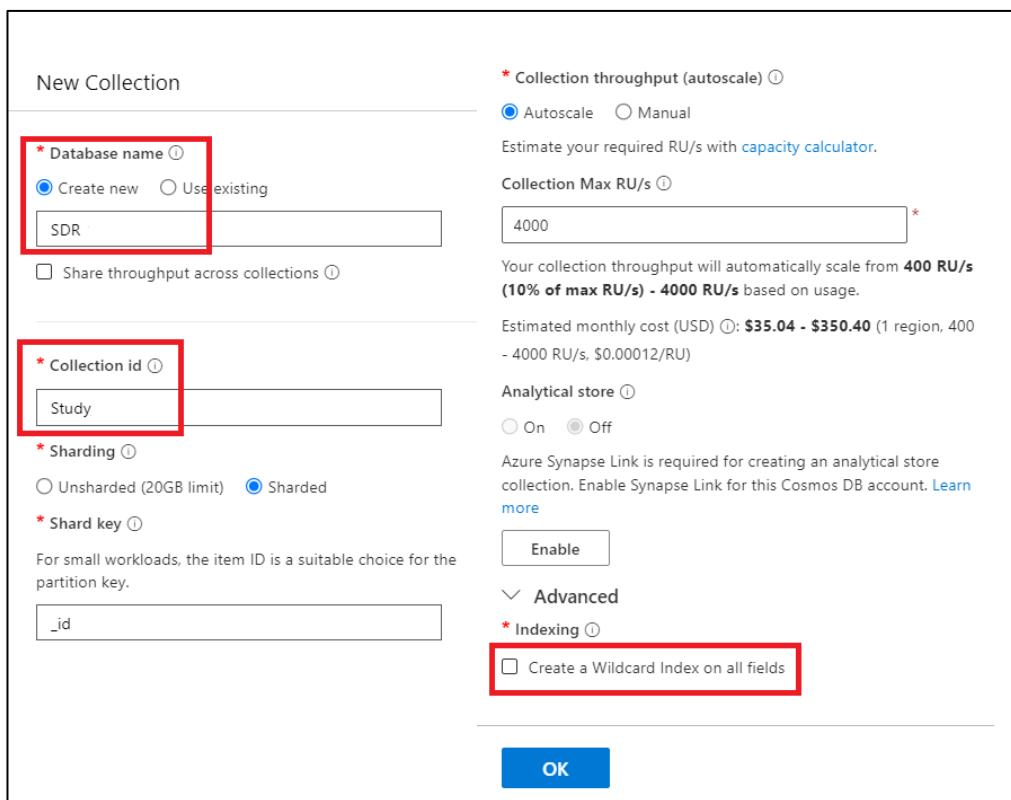


Figure 17 Creating Collection in CosmosDB



New Collection

* Database name ⓘ
 Create new Use existing

Share throughput across collections ⓘ

* Collection id ⓘ

* Sharding ⓘ
 Unsharded (20GB limit) Sharded

* Shard key ⓘ
For small workloads, the item ID is a suitable choice for the partition key.

* Collection throughput (autoscale) ⓘ
 Autoscale Manual
Estimate your required RU/s with [capacity calculator](#).

Collection Max RU/s ⓘ
 *

Your collection throughput will automatically scale from **400 RU/s (10% of max RU/s)** - **4000 RU/s** based on usage.

Estimated monthly cost (USD) ⓘ: **\$35.04 - \$350.40** (1 region, 400 - 4000 RU/s, \$0.00012/RU)

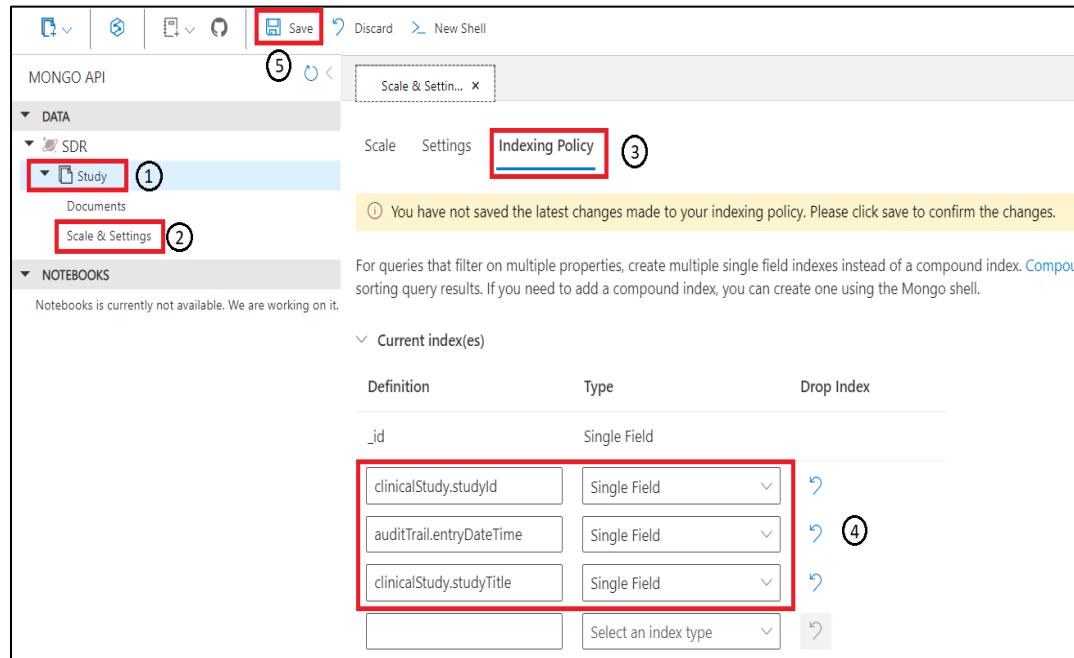
Analytical store ⓘ
 On Off
Azure Synapse Link is required for creating an analytical store collection. Enable Synapse Link for this Cosmos DB account. [Learn more](#)

✓ Advanced

* Indexing ⓘ
 Create a Wildcard Index on all fields

iv. Add Single Field indexes as mentioned below for Study collection.

Figure 18 CosmosDB Indexing



2.8.2. PaaS Setup – 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:

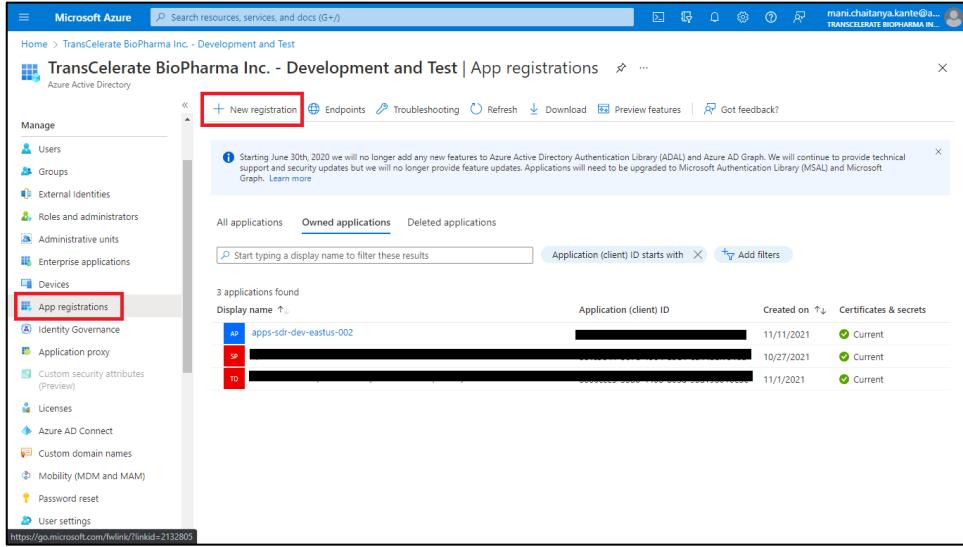
- Global Administrator level of access at Active Directory level.

STEPS

SDR UI APP REGISTRATION:

- Go To Azure AD → App Registrations → New Registration

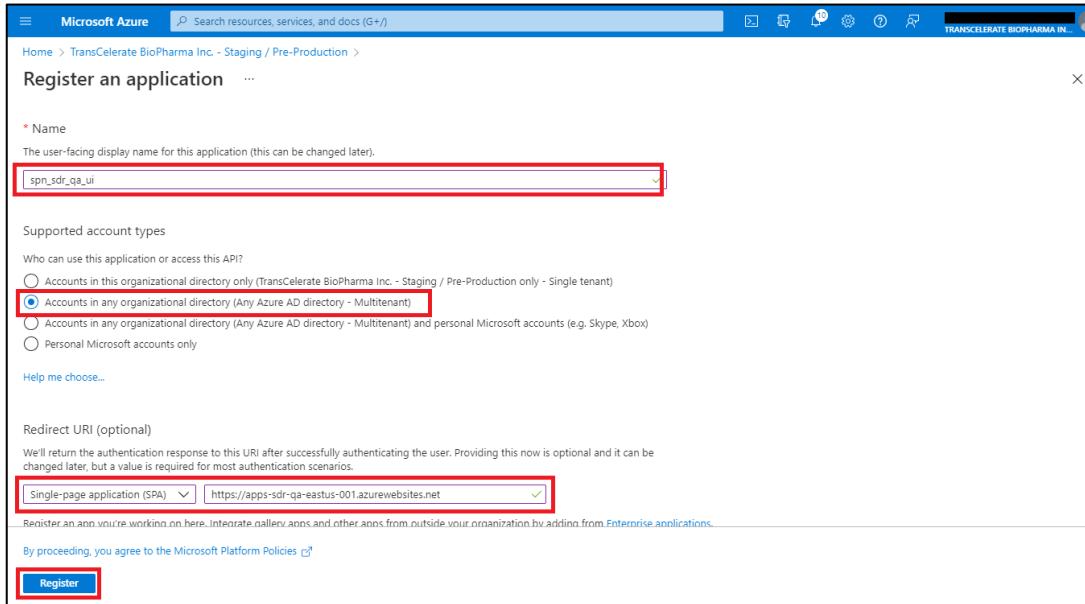
Figure 19 Azure AD - App Registration



The screenshot shows the Microsoft Azure Active Directory App registrations page. The left sidebar has a 'Manage' section with various options like Users, Groups, External Identities, etc., and 'App registrations' is selected and highlighted with a red box. The main area shows a table of registered applications. A new application entry is being created, indicated by the 'New registration' button at the top left of the table area, which is also highlighted with a red box.

- ii. Provide Name, and in Supported Account Types, select “Accounts in any organizational directory – Multitenant” and enter Redirect URL which is the Backend Application URL)
- iii. Click on Register to save the App registration.

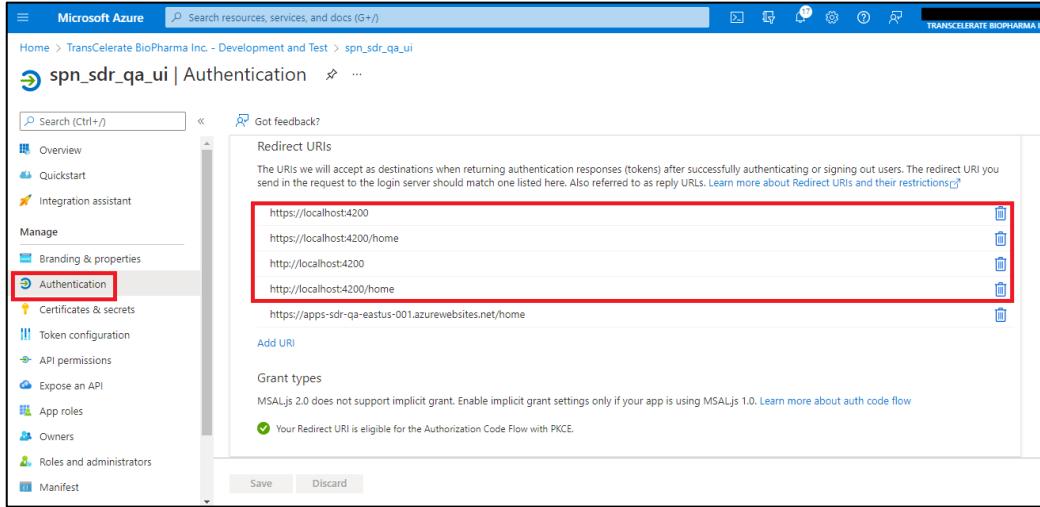
Figure 20 Azure AD - Register Application



The screenshot shows the 'Register an application' form. The 'Name' field is filled with 'spn_sdr_qa_ui' and is highlighted with a red box. In the 'Supported account types' section, the 'Accounts in any organizational directory (Any Azure AD directory - Multitenant)' option is selected and highlighted with a red box. Other options like 'Accounts in this organizational directory only' and 'Personal Microsoft accounts only' are also shown. The 'Register' button at the bottom is also highlighted with a red box.

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

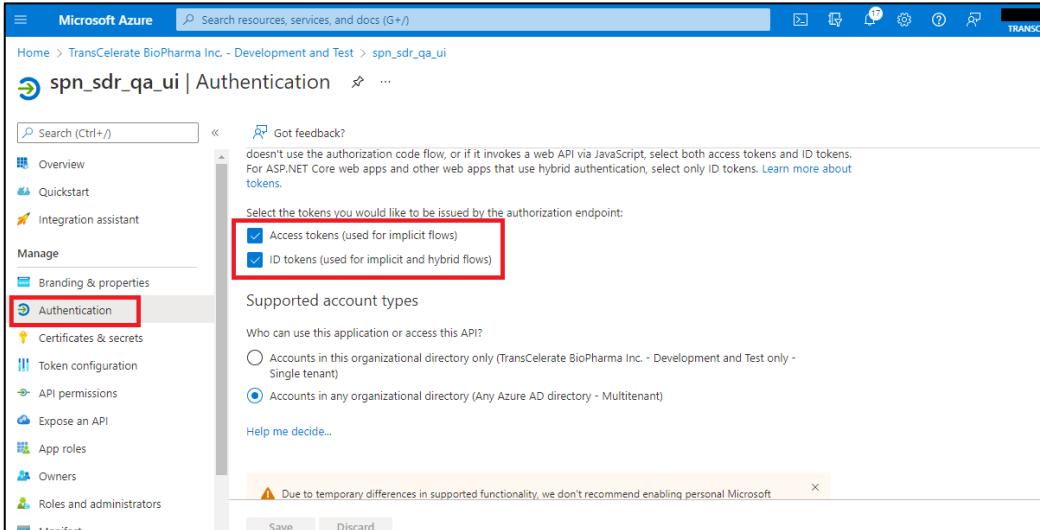
Figure 21 Azure AD - App Registration - Redirect URI



The screenshot shows the Microsoft Azure portal's 'Authentication' blade for an app named 'spn_sdr_qa_ue'. The 'Manage' sidebar on the left has 'Authentication' selected. The main area shows the 'Redirect URIs' section with four entries: https://localhost:4200, https://localhost:4200/home, http://localhost:4200, and http://localhost:4200/home. A tooltip above the list explains that these URLs will be accepted as destinations for authentication responses. Below the list, it says 'Grant types' and 'MSAL.js 2.0 does not support implicit grant. Enable implicit grant settings only if your app is using MSAL.js 1.0.' A green checkmark indicates that the redirect URI is eligible for Authorization Code Flow with PKCE.

v. Check to select “Access tokens” and “ID tokens”

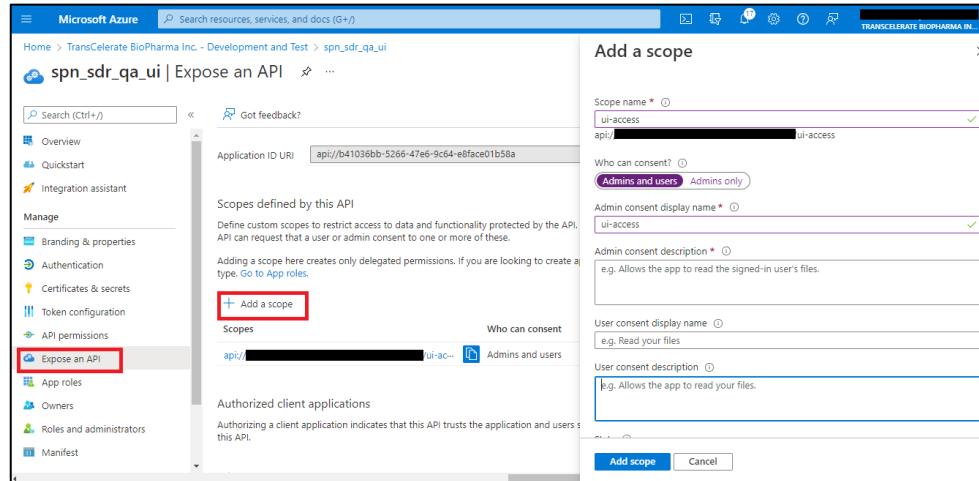
Figure 22 Azure AD - App Registration - Authentication



The screenshot shows the Microsoft Azure portal's 'Authentication' blade for the same app. The 'Manage' sidebar has 'Authentication' selected. The main area has a note about token selection: 'Select the tokens you would like to be issued by the authorization endpoint' with 'Access tokens (used for implicit flows)' and 'ID tokens (used for implicit and hybrid flows)' checked. A tooltip above the note explains that this applies to web API via JavaScript or hybrid authentication. Below the note, it lists 'Supported account types' with the option 'Accounts in any organizational directory (Any Azure AD directory - Multitenant)' selected. A warning message at the bottom notes 'Due to temporary differences in supported functionality, we don't recommend enabling personal Microsoft accounts.'

vi. 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 23 Azure AD - App Registration - Expose an API



- vii. 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 24 Azure AD - App Registration - API Permission

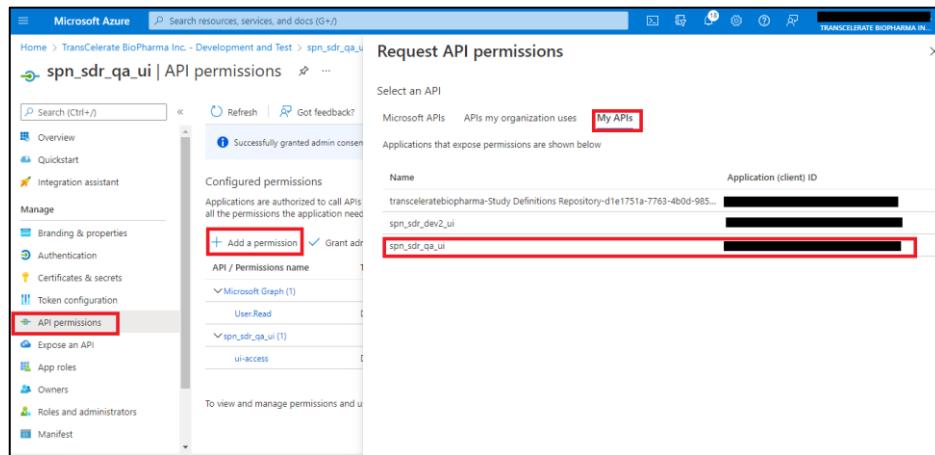
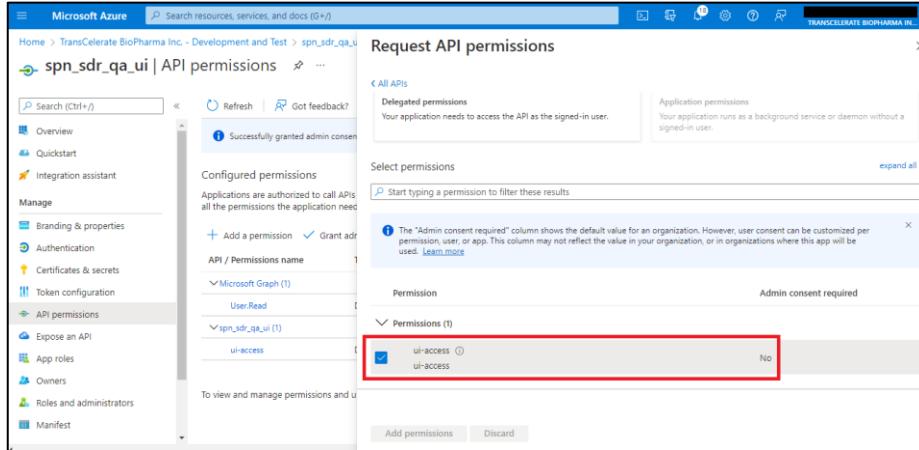
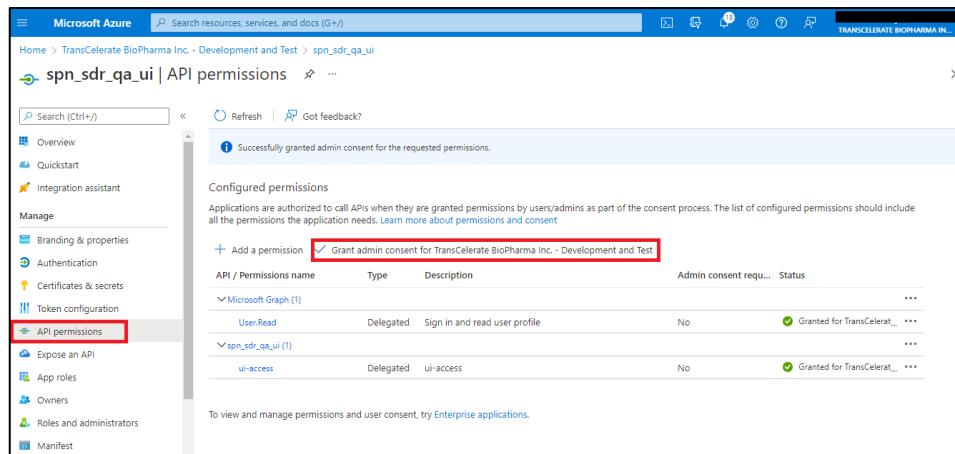


Figure 25 Azure AD - App Registration - Add API Permission



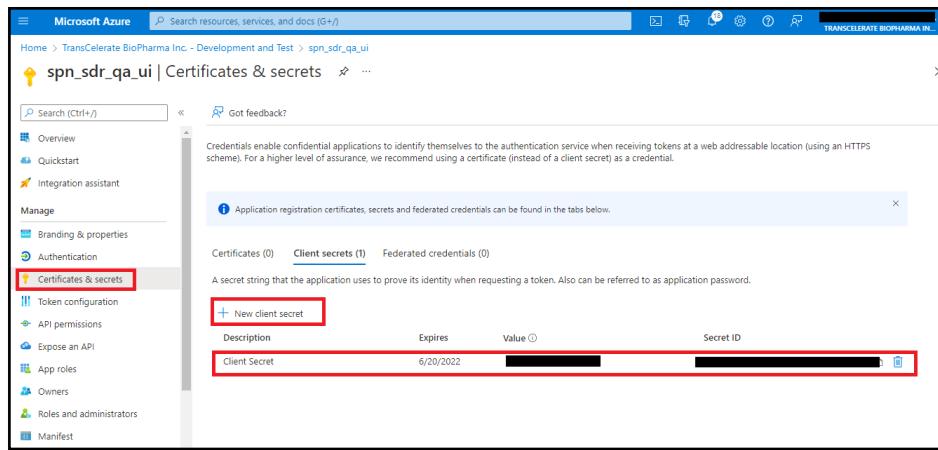
viii. Grant Admin consent.

Figure 26 Azure AD - App Registration - API Permission - Grant Admin Consent



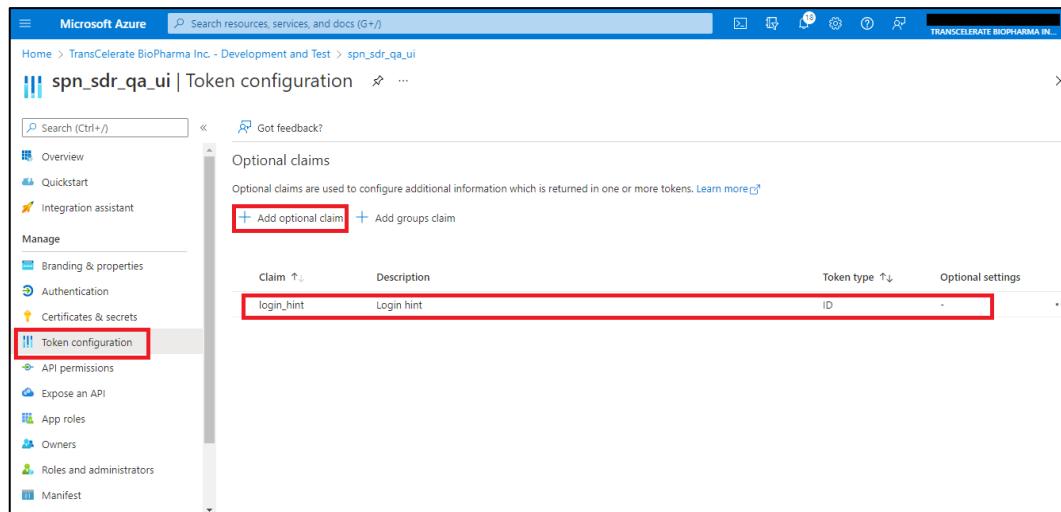
ix. Go to certificates & secrets → click on client secrets → click new client secret and copy the value and add it on Key Vault secrets.

Figure 27 Azure AD - App Registration - Certificates & Secrets



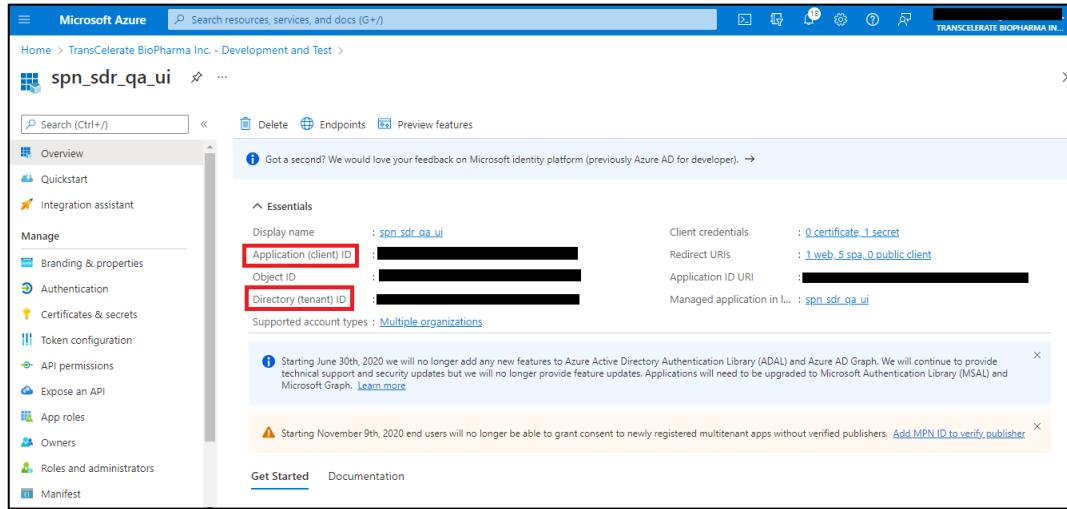
x. Go to Token Configuration and click “Add optional Claim” to add “Login_hint”.

Figure 28 Azure AD - App Registration - Token Configuration



xi. 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 2.8.2 for Key Vault secrets.

Figure 29 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.

2.8.3. UI App Path Mapping

GOAL:

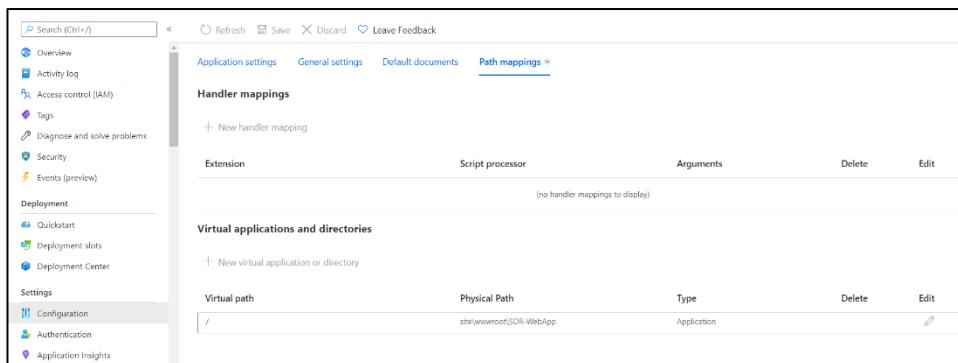
Update the Path Mapping of UI Azure App Service.

PRE-REQUISITES:

- Contributor access at Resource Group level.

STEPS:

- Navigate to UI App Service instance in App resource group.
- Go to Configuration → Path Mappings
- Change the Physical Path from site\wwwroot to site\wwwroot\SDR-WebApp for accessing the app URL.



2.8.4. API Management

GOAL:

Create API Management Base URL for accessing the API's.

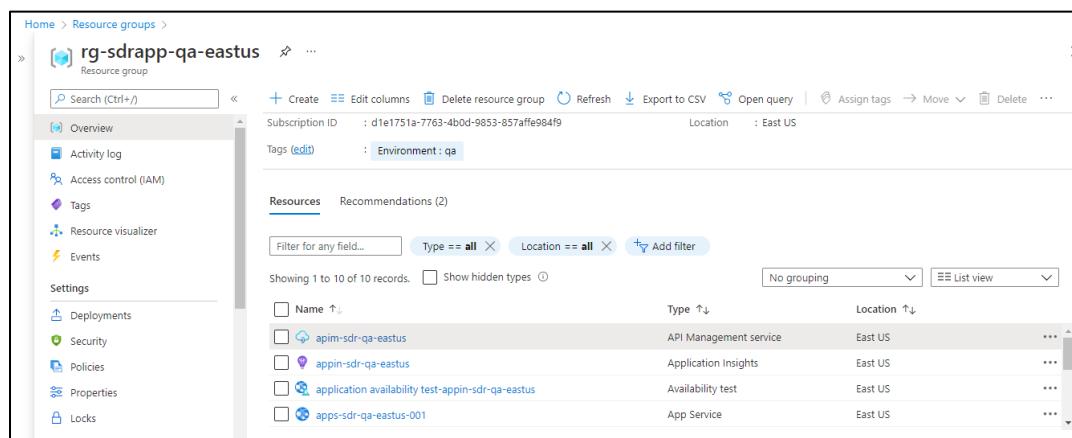
PRE-REQUISITES:

- Contributor access at Resource Group level.

STEPS:

- Navigate to APIM instance in App resource group.

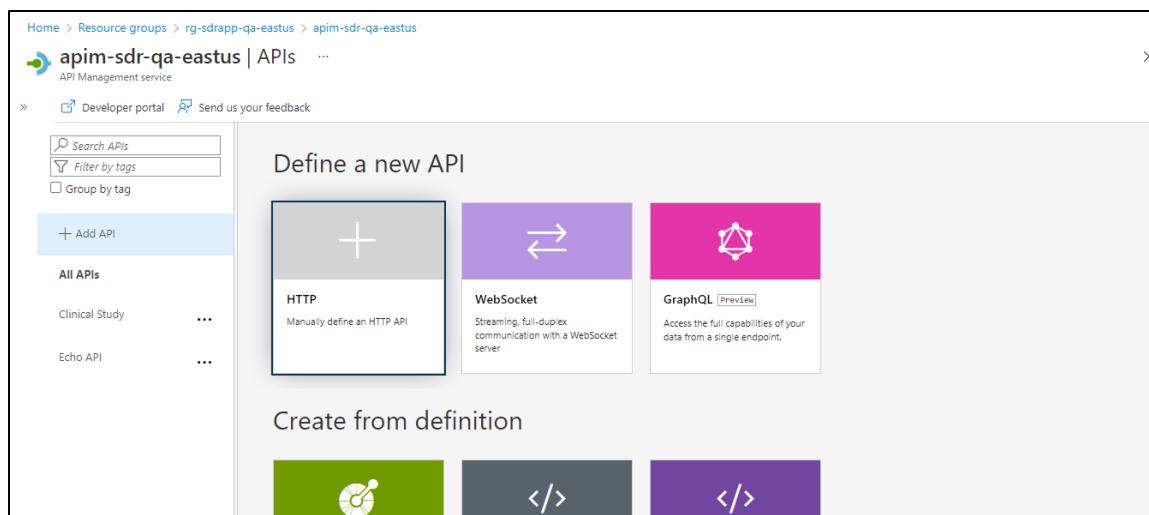
Figure 30 Azure Portal - SDR App Resource Group



Name	Type	Location	Actions
apim-sdr-qa-eastus	API Management service	East US	...
appin-sdr-qa-eastus	Application Insights	East US	...
application availability test-appin-sdr-qa-eastus	Availability test	East US	...
apps-sdr-qa-eastus-001	App Service	East US	...

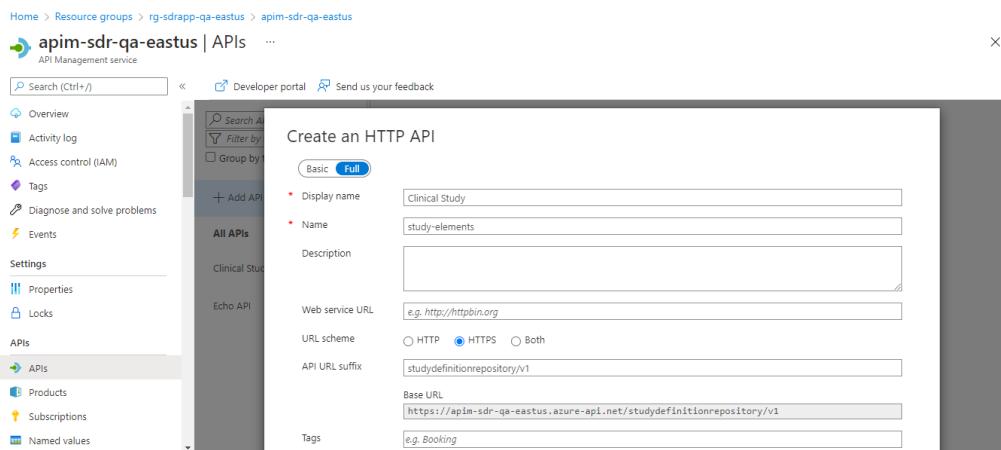
- In the left navigation of your API Management instance, select APIs > select HTTP API tile.

Figure 31 API Management - Add API



- vi. In the “Create an HTTP API” specification window, select “Full” and provide mandatory details like name. This entry corresponds to a single API endpoint in SDR.

Figure 32 API Management - Create an HTTP API



2.8.5. 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.

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 2.8.2 step 6 for scope URL (E.g.: api://0000-0000-0000-0000/ui-access)
AzureAd-Authority	Provide the Azure Ad authority value (https://login.microsoftonline.com/ (Provide Azure AD Tenant ID))

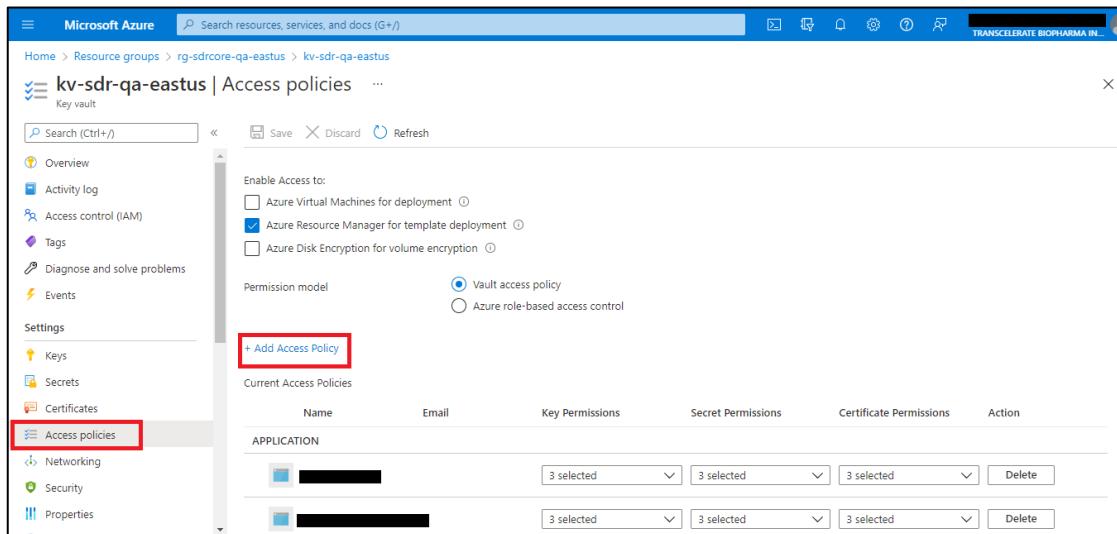
AzureAd-ClientId	Provide the App Registration client ID, refer to Section 2.8.2 App Registration step 11 for client ID
AzureAd-LoginUrl	Provide the Front End (UI) App Service URL.
AzureAd-RedirectUrl	Provide the Redirect URL (E.g.: https://Front End App service URL (UI)/home)
AzureAd-TenantId	Provide the Azure AD Tenant ID, refer to Section 2.8.2 App Registration step 11 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.

STEPS FOR ADDING ACCESS POLICY:

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

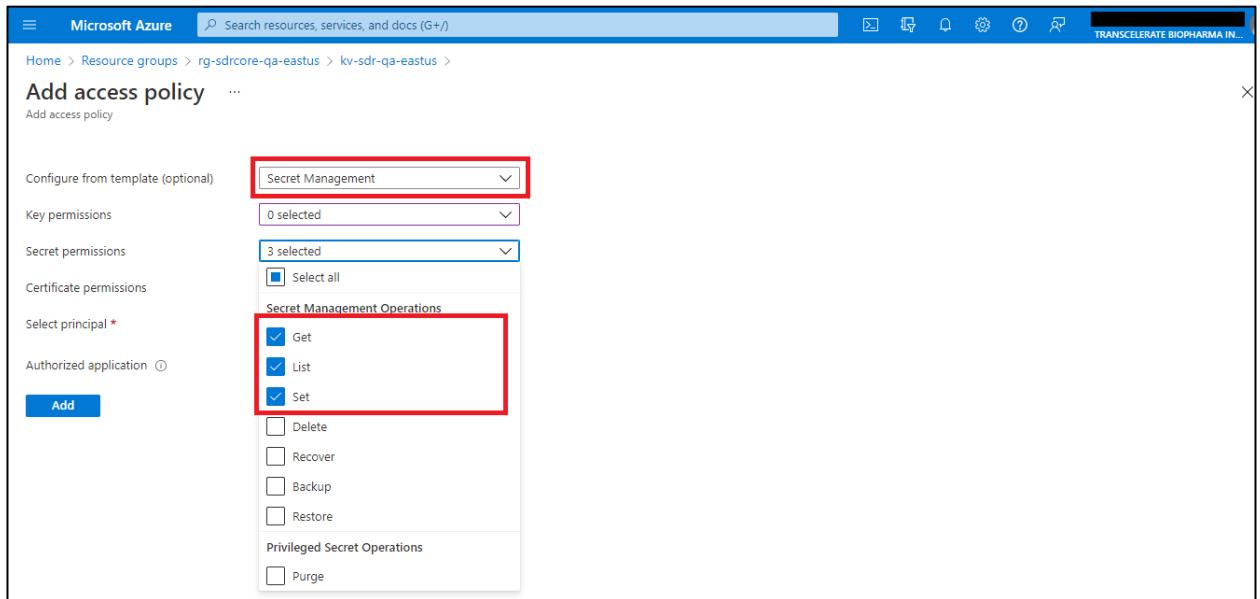
- Go to → Key Vault → Select Access Policies → Click on Add Access Policy

Figure 33 Add Key Vault Access Policy



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

Figure 34 Select Secret Permissions in Access Policy in Key Vault



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

Figure 35 Select Principal (List of users) In Key Vault Access Policy

Add access policy ...

Add access policy

Configure from template (optional)

Key permissions

Secret permissions

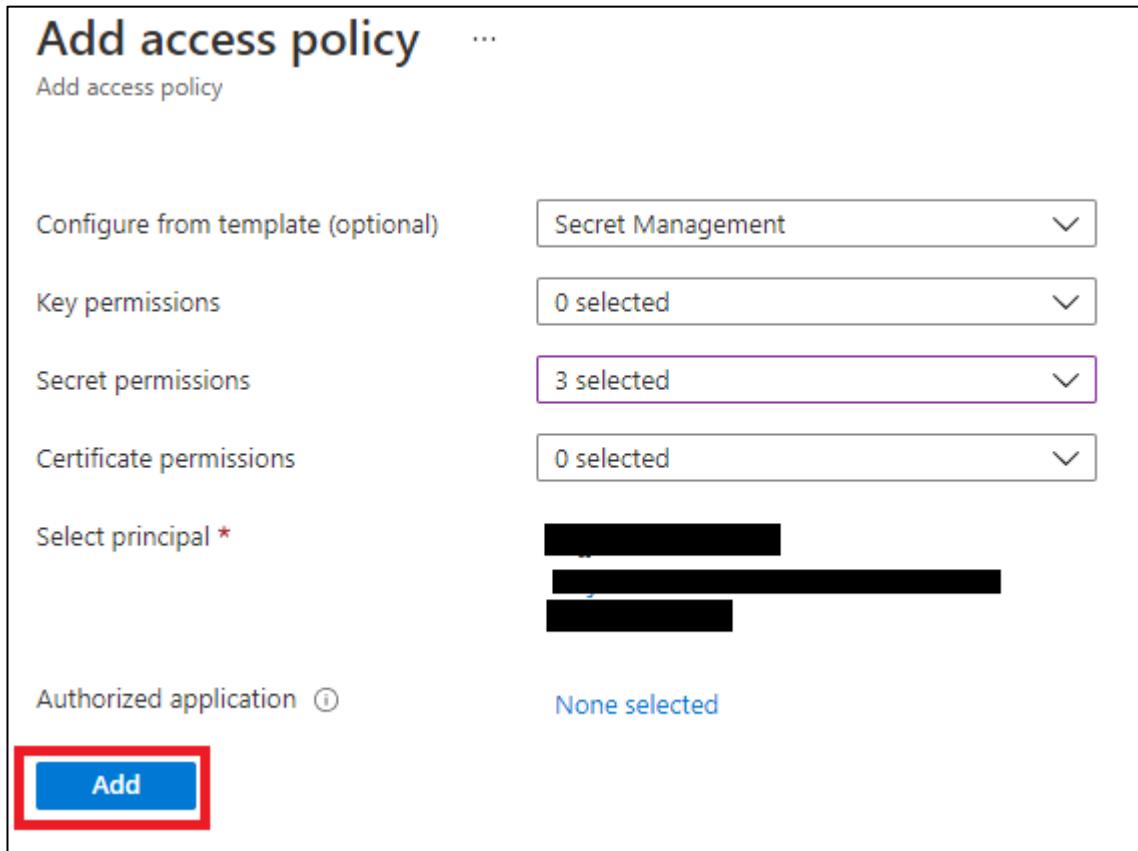
Certificate permissions

Select principal *

Authorized application ⓘ

None selected

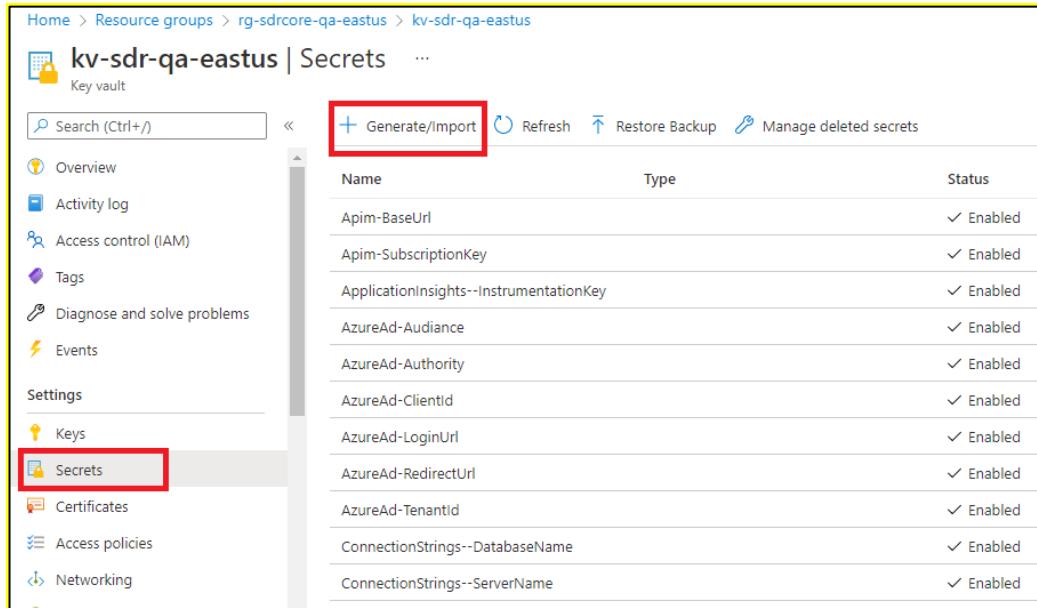
Add

**STEPS FOR ADDING KEY VAULT SECRETS:**

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

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

Figure 36 Create Secrets in Key Vault



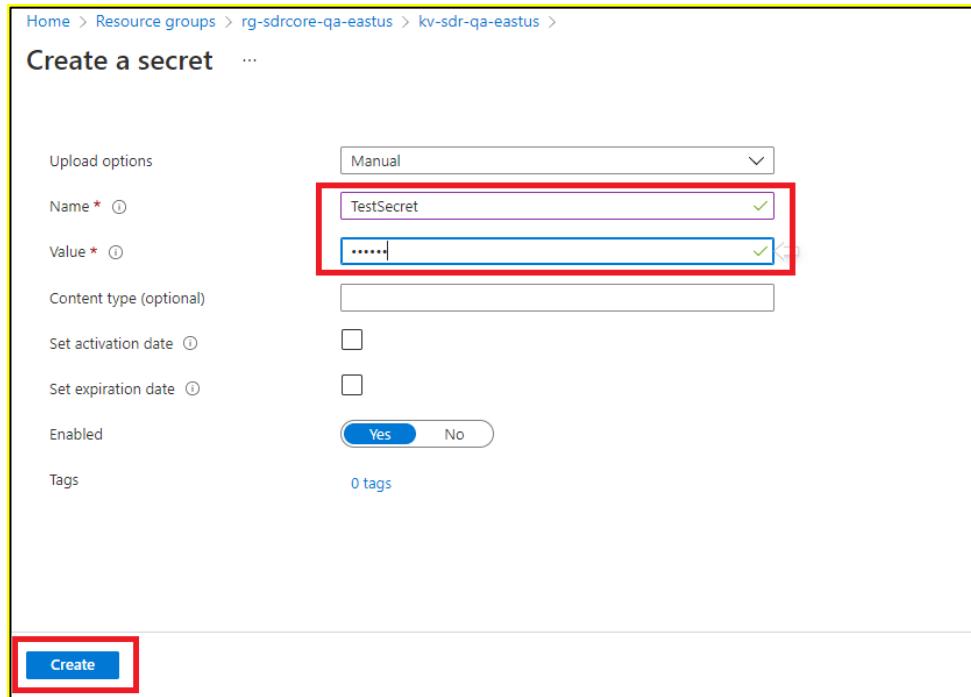
The screenshot shows the 'kv-sdr-qa-eastus | Secrets' page in the Azure portal. The left sidebar includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings (with Keys and Secrets selected), Certificates, Access policies, and Networking. The main area displays a table of 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
.ConnectionStrings--DatabaseName		✓ Enabled
.ConnectionStrings--ServerName		✓ Enabled

A red box highlights the 'Generate/Import' button at the top right of the page.

ii. Provide Secret Name and Value → Select Create

Figure 37 Add Secrets values in Key Vault



The screenshot shows the 'Create a secret' dialog. The 'Name' field is set to 'TestSecret' and the 'Value' field contains '.....'. Other fields include 'Content type (optional)', 'Set activation date', 'Set expiration date', 'Enabled' (set to 'Yes'), and 'Tags' (0 tags). A red box highlights the 'Create' button at the bottom left.

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 settings and configurations that have been configured on Terraform IaC code.



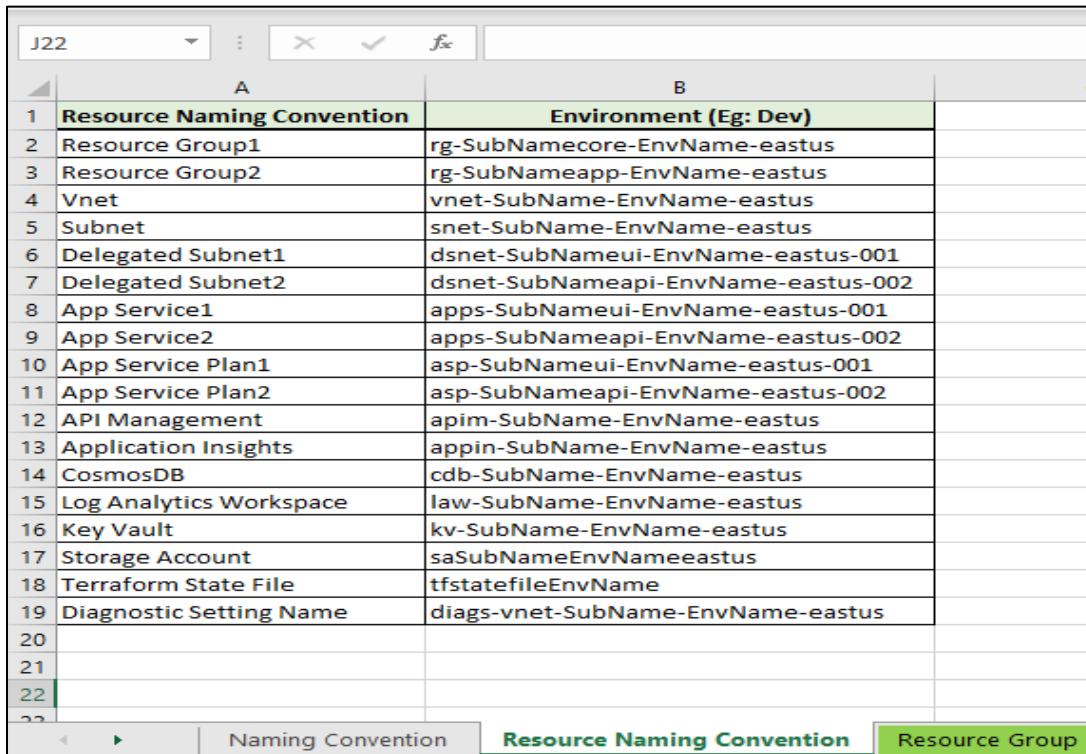
Naming Convention followed for all the resources is as below -

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

Figure 38 Resource Naming Convention

Naming Convention	<pre><ResourceType>-<App/Svc>-<Purpose/Segment/Environment>-<Region> ####</pre> <hr/> <p style="margin-top: 10px;"><ResourceType> = Mandatory (upto 4 Characters) <App/Svc> = Optional (upto 5 Characters) <Purpose/Segment/Environment> = Mandatory (Unlimited) <Region> = Optional (upto 6 Characters) <####> = Optional (Instance Number or level # = 3 Characters)</p>	
		

Figure 39 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 40 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 Sett	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

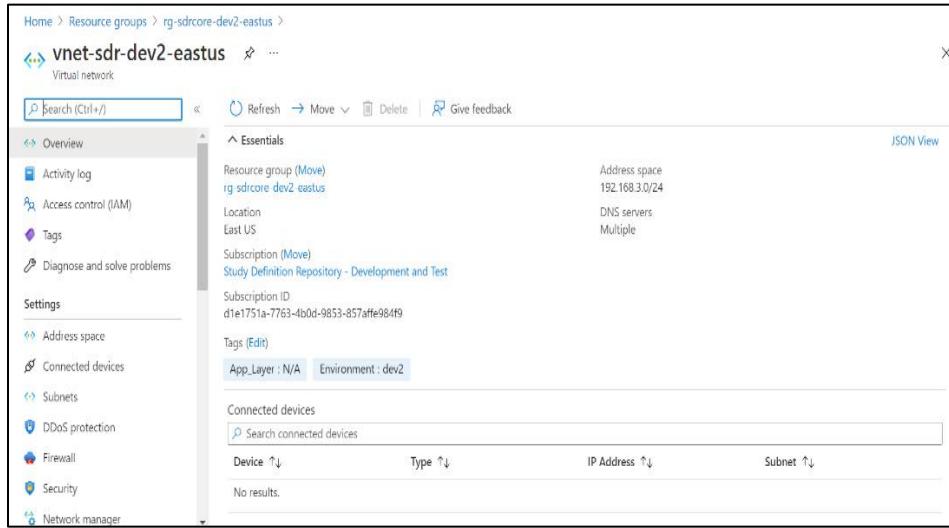
PRE-REQUISITES:

- 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 41 Virtual Network



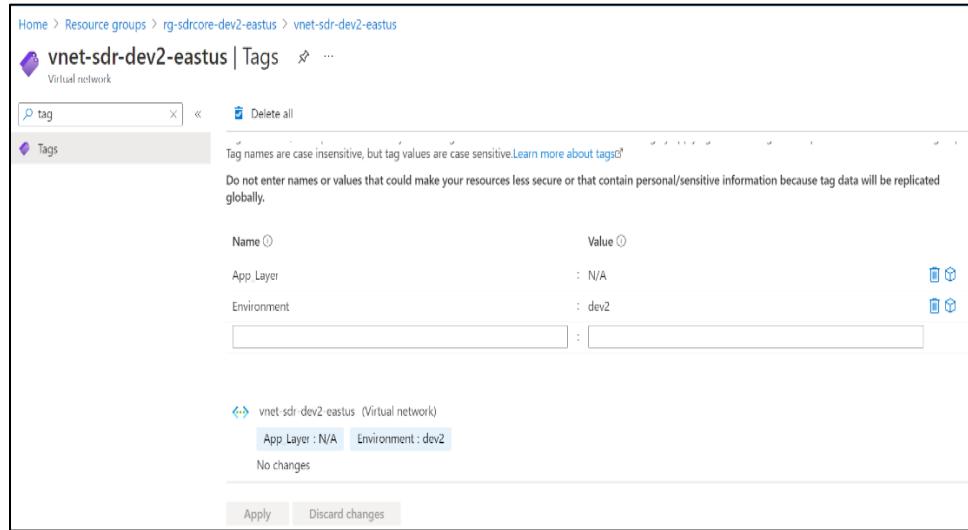
- vi. Go to Security tab and verify that Bastion Host is set as per the LLD
- vii. Go to DDoS Protection tab and verify that it is set as per the LLD

Figure 42 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

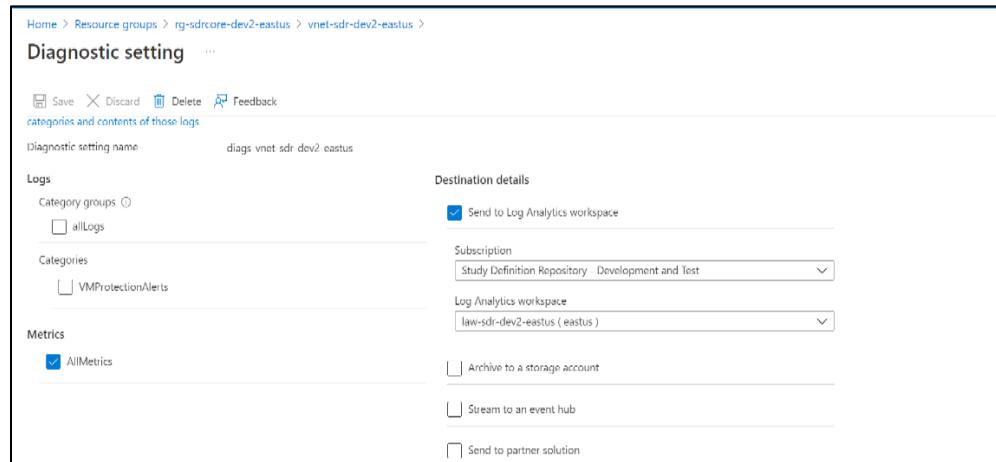
Figure 43 Virtual Network - Tags



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 44 Virtual Network - Diagnostic Setting



3.3. Subnet

Validation of Virtual Subnet configuration.

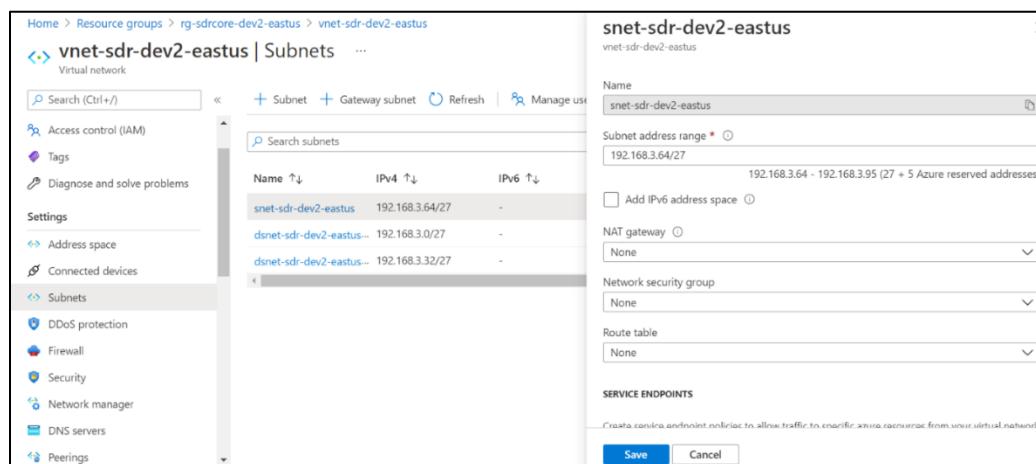
PRE-REQUISITES:

- vii. Reader level of access at Resource group level
- viii. 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 45 Subnet Details



The screenshot shows the Azure portal interface for managing subnets. On the left, there's a sidebar with various settings like Address space, Connected devices, Subnets (which is selected), DDoS protection, Firewall, Security, Network manager, DNS servers, and Peering. The main area shows a list of subnets under the heading 'vnet-sdr-dev2-eastus | Subnets'. The list includes:

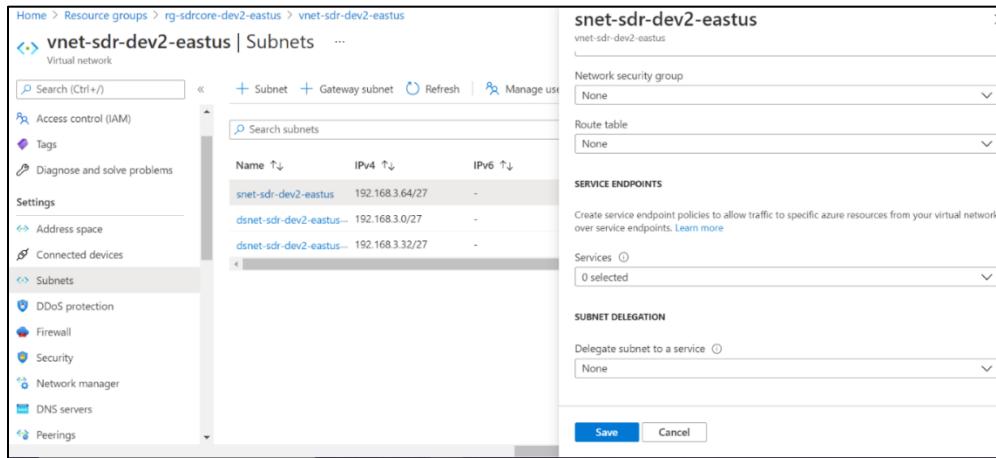
Name	IPv4	IPv6
snet-sdr-dev2-eastus	192.168.3.64/27	-
dsnet-sdr-dev2-eastus...	192.168.3.0/27	-
dsnet-sdr-dev2-eastus...	192.168.3.32/27	-

To the right, a detailed configuration pane is open for the 'snet-sdr-dev2-eastus' subnet. It shows the following details:

- Name:** snet-sdr-dev2-eastus
- Subnet address range:** 192.168.3.64 - 192.168.3.95 (27 + 5 Azure reserved addresses)
- Add IPv6 address space:** (checkbox)
- NAT gateway:** (radio button set to 'None')
- Network security group:** (dropdown set to 'None')
- Route table:** (dropdown set to 'None')
- SERVICE ENDPOINTS:** (Note: Create a service endpoint or use an existing one to allow traffic to specific services from your virtual machine.)

At the bottom of the configuration pane are 'Save' and 'Cancel' buttons.

Figure 46 Subnet Details



3.4. Delegated Subnet

Validation of Delegated Subnet configuration.

PRE-REQUISITES:

- Reader level of access at Resource group Level.

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 47 Delegated Subnet-001 Details

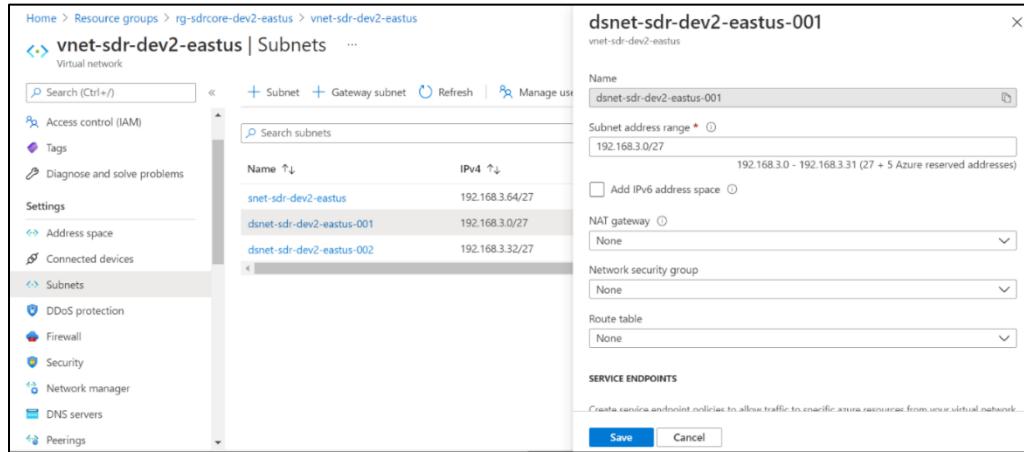


Figure 48 Delegated Subnet-001 Service Endpoints Details

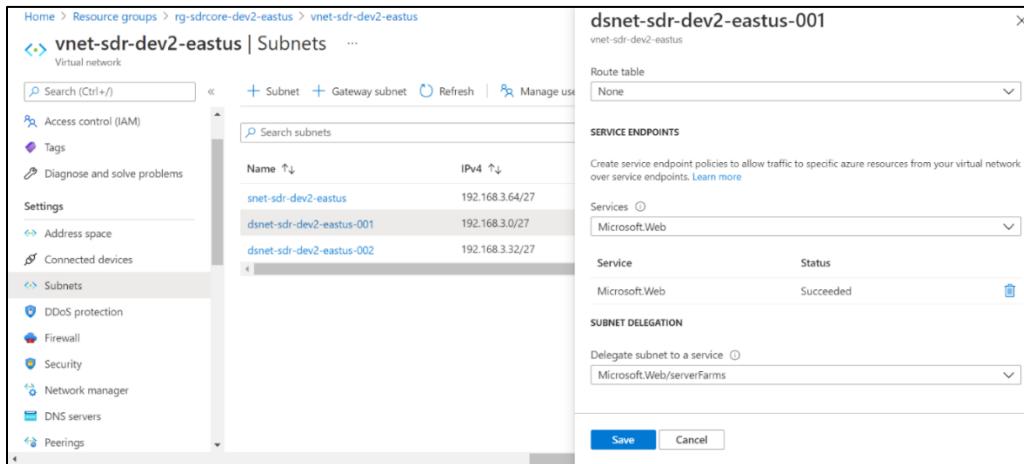


Figure 49 Delegated Subnet-002 Details

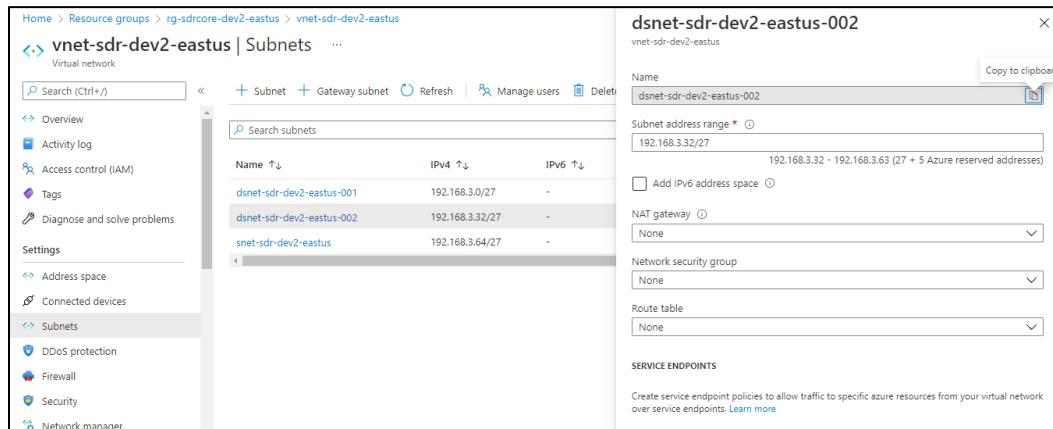
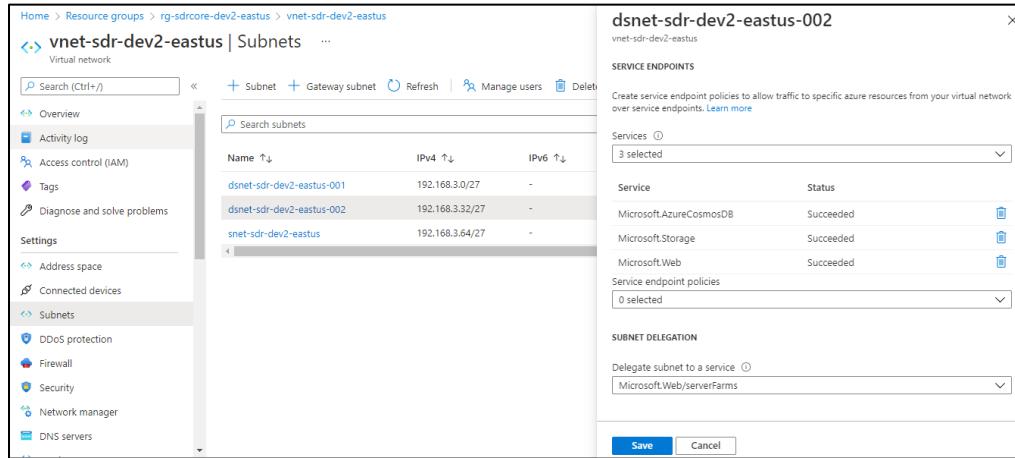


Figure 50 Delegated Subnet-002 Service Endpoints Details



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

4. Application Code Deployment

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

Step to be followed:

- 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 App Service for SDR API and copy the name from Overview blade. This will be the AZURE_WEBAPP_NAME for ddf-sdr-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 ddf-sdr-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 ddf-sdr-api → Settings → Secrets → Actions and add/replace values for AZURE_SP, AZURE_WEBAPP_NAME and KEYVAULT_NAME by clicking on update.
- Login to GitHub and navigate to ddf-sdr-ui → Settings → Secrets → Actions and add/replace values for AZURE_SP, AZURE_WEBAPP_NAME and KEYVAULT_NAME by clicking on Update.

4.2 Deploy the UI Application

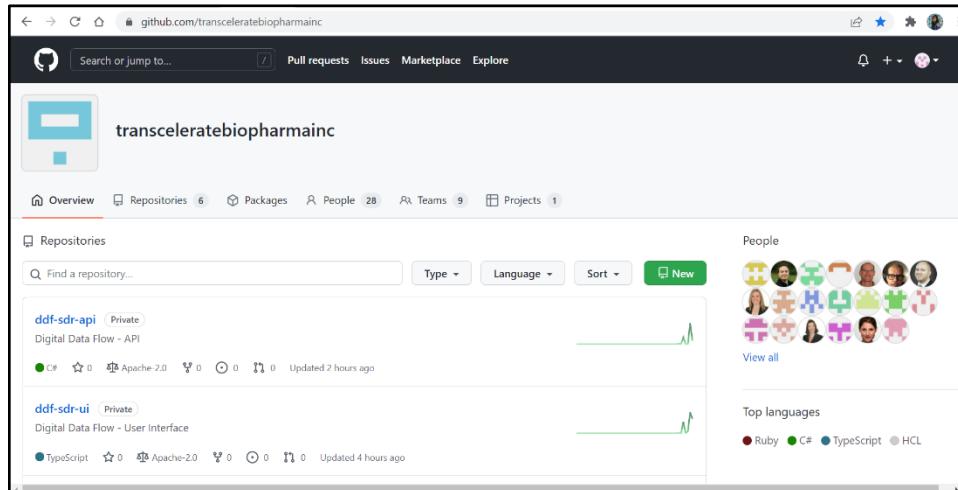
PRE-REQUISITES

- Contributor level of access at Resource Group level.

DEPLOYMENT STEPS:

- Go to <https://github.com/transceleratebiopharmainc>, GitHub URL.

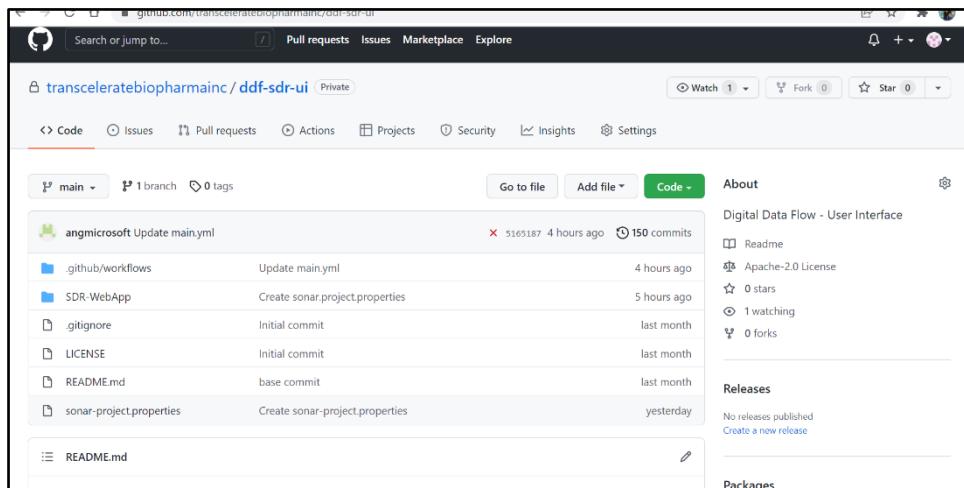
Figure 51 TransCelerate GitHub Project



- Select the required repository, here ddf-sdr-ui.URL –

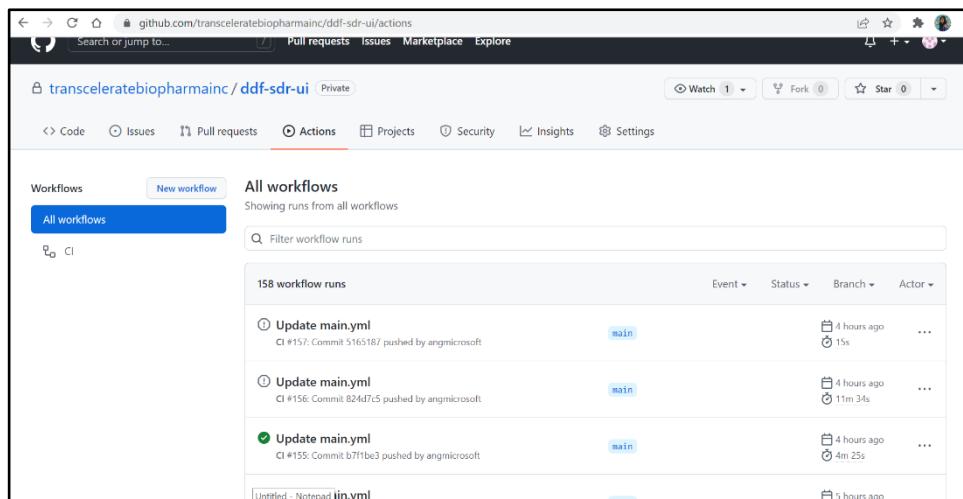
<https://github.com/transceleratebiopharmainc/ddf-sdr-ui>

Figure 52 TransCelerate GitHub SDR UI Repo



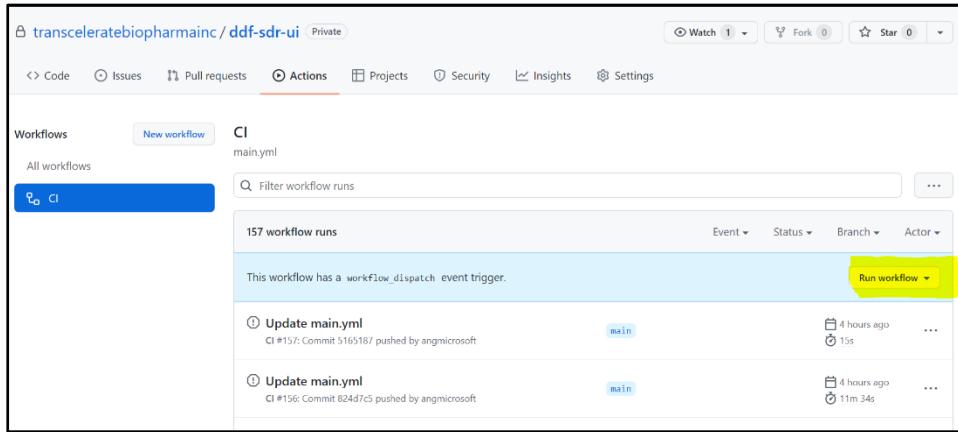
iii. Click on Actions tab.

Figure 53 SDR UI Repo - GitHub Actions



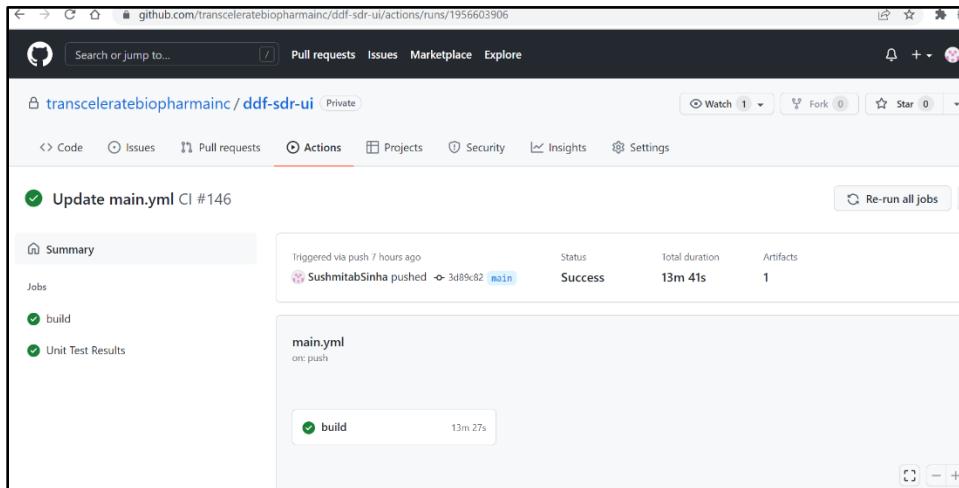
iv. Click on the workflow CI under All workflow.

Figure 54 GitHub Actions - CI Workflow



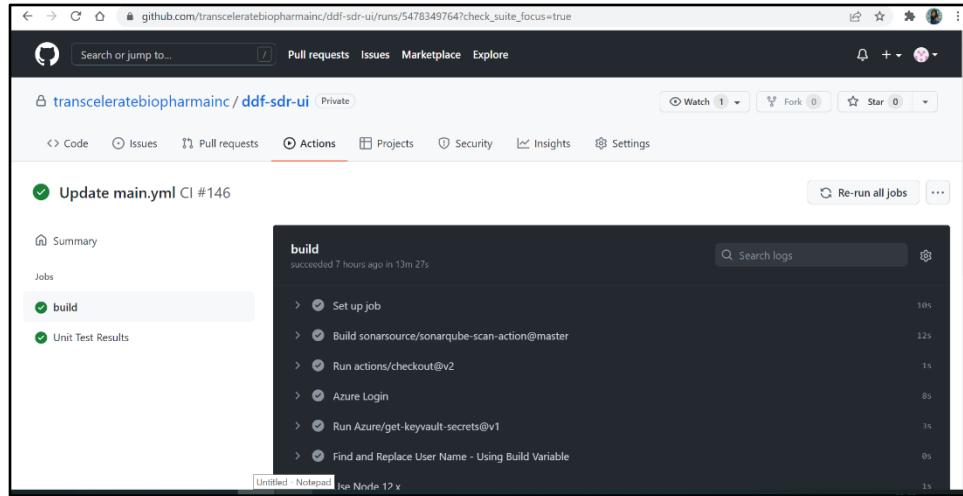
- v. Click on Run Workflow on the right-hand side. Then the action will be triggered.

Figure 55 GitHub - CI Workflow Run



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

Figure 56 GitHub CI Workflow Output



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

4.3 Deploy Back-End API

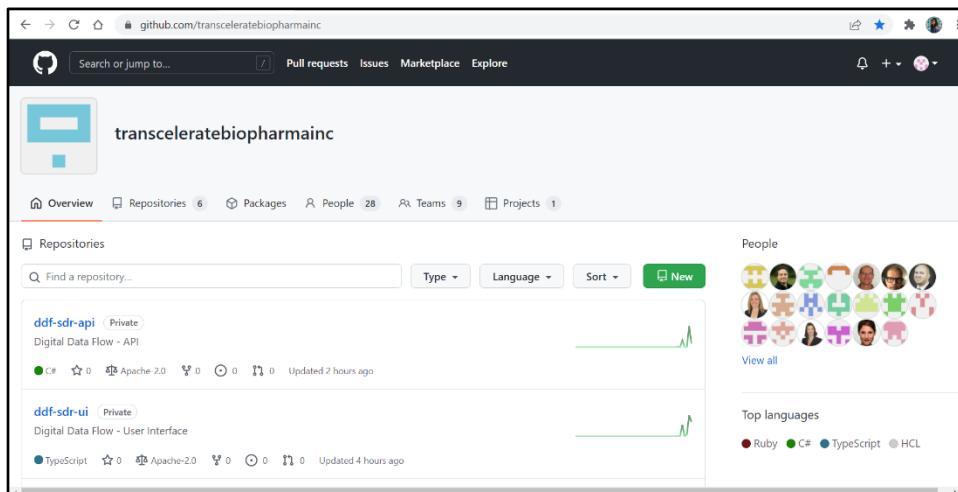
PRE-REQUISITES

- Contributor access at Resource group Level.

DEPLOYMENT STEPS:

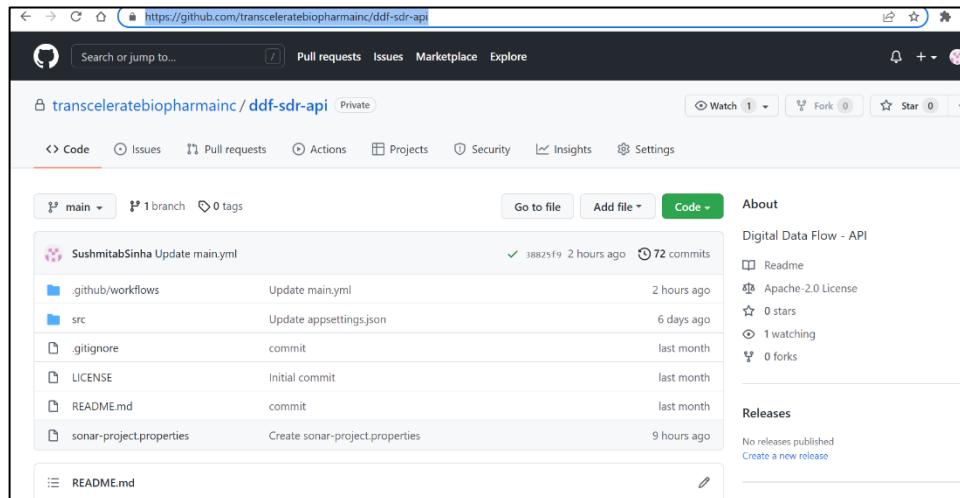
- Go to [GitHub URL](https://github.com/transceleratebiopharmainc).

Figure 57 TransCelerate GitHub Project



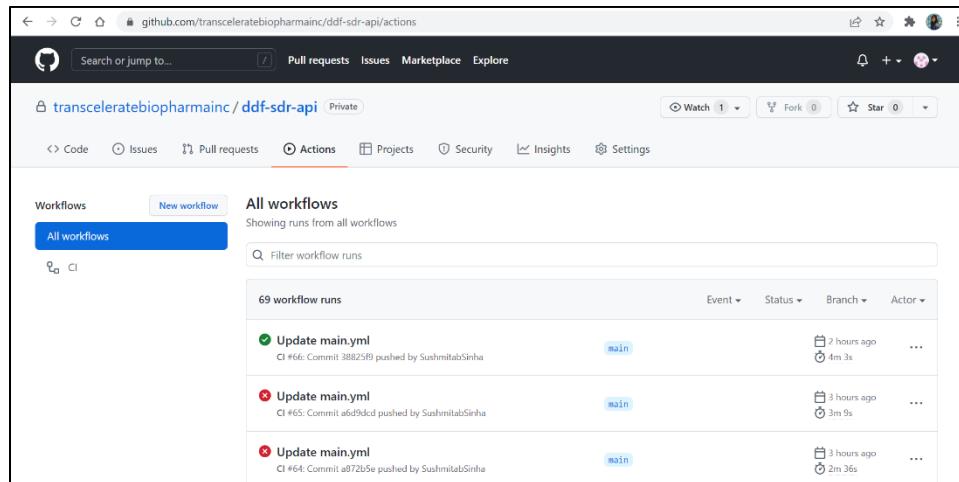
- ii. Select the required repository, here **ddf-sdr-api**.
URL - <https://github.com/transceleratebiopharmainc/ddf-sdr-api>

Figure 58 TransCelerate GitHub SDR API Repo



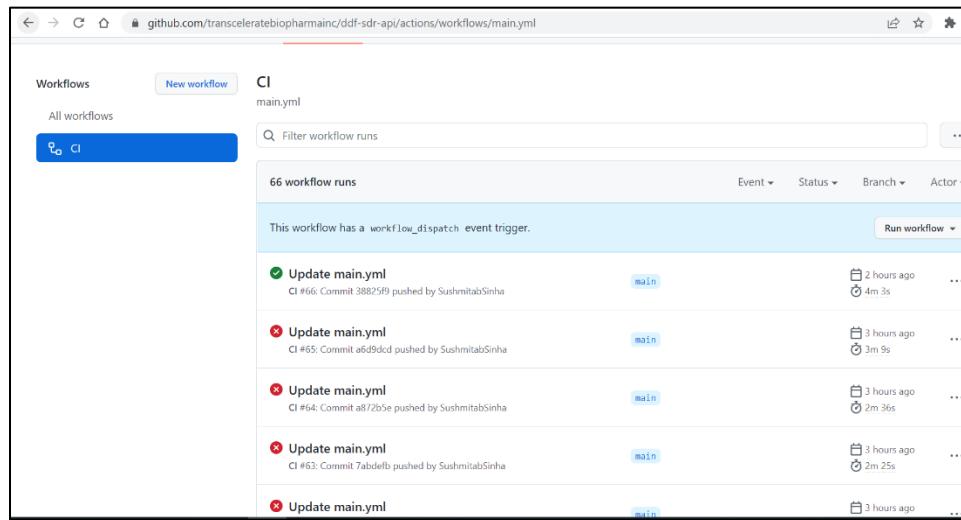
- iii. Click on Actions tab.

Figure 59 GitHub Actions CI Workflow



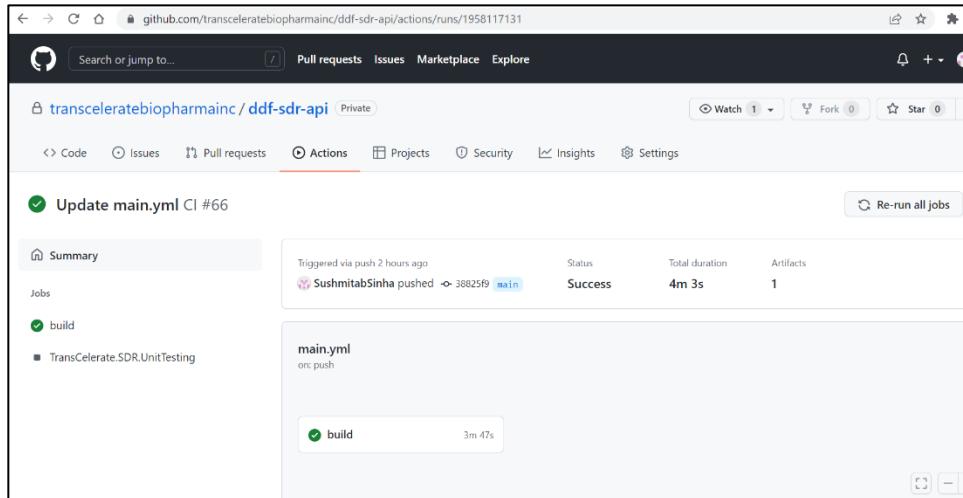
- iv. Click on the workflow CI under All workflow.

Figure 60 GitHub CI Workflow Run



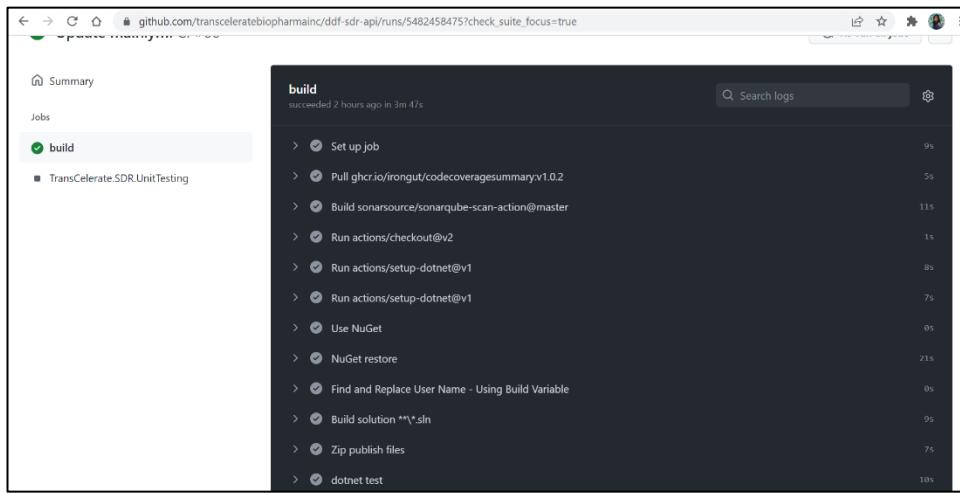
- v. Click on Run Workflow on the right-hand side. Then the action will be triggered.

Figure 61 GitHub CI Workflow Run



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

Figure 62 GitHub CI Workflow Output



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

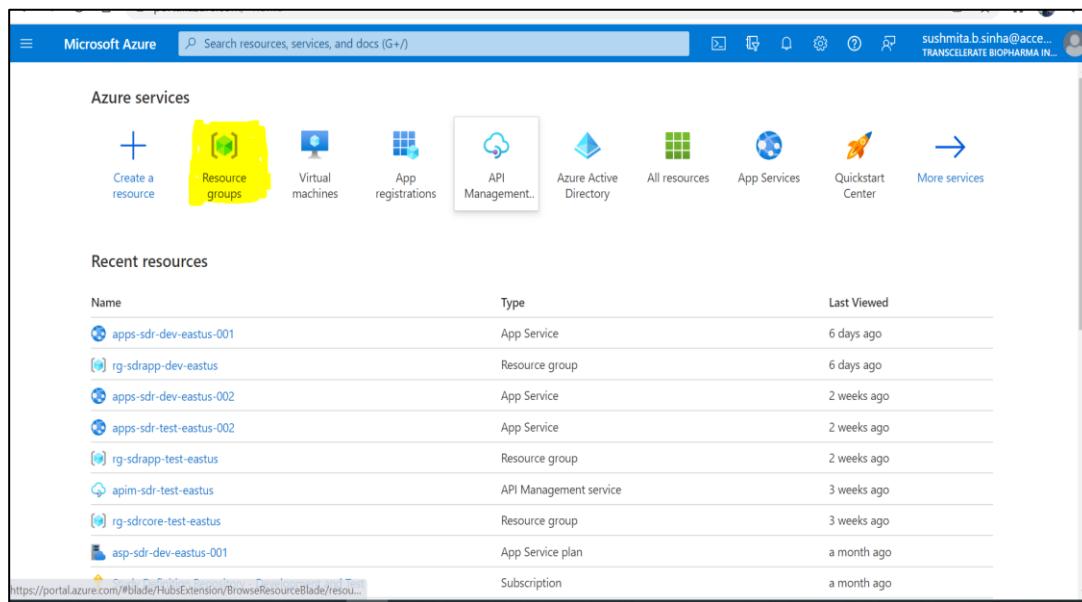
4.4 Deployment Verification

UI APPLICATION VERIFICATION STEPS:

This is to verify the UI Application deployment was successful.

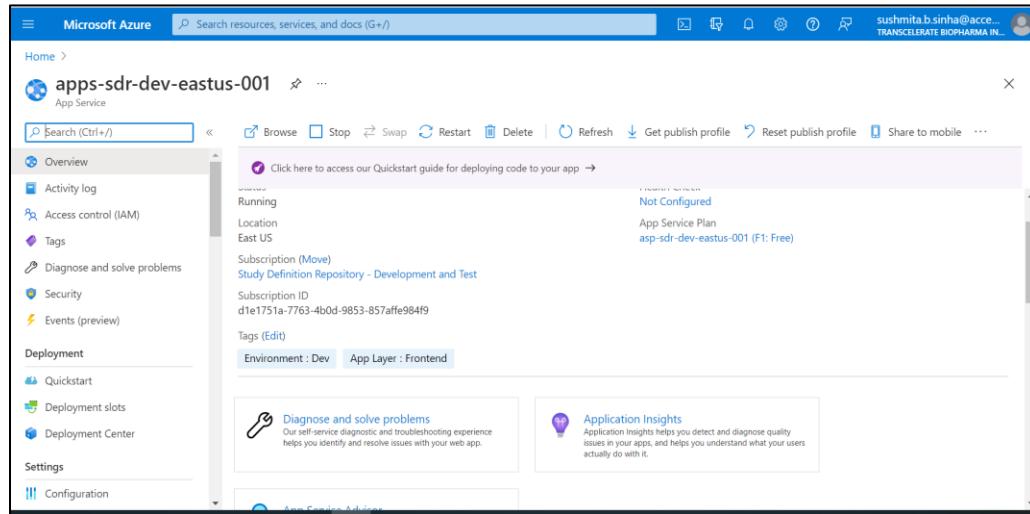
- Go to portal.azure.com. Click on resource group.

Figure 63 Azure Portal



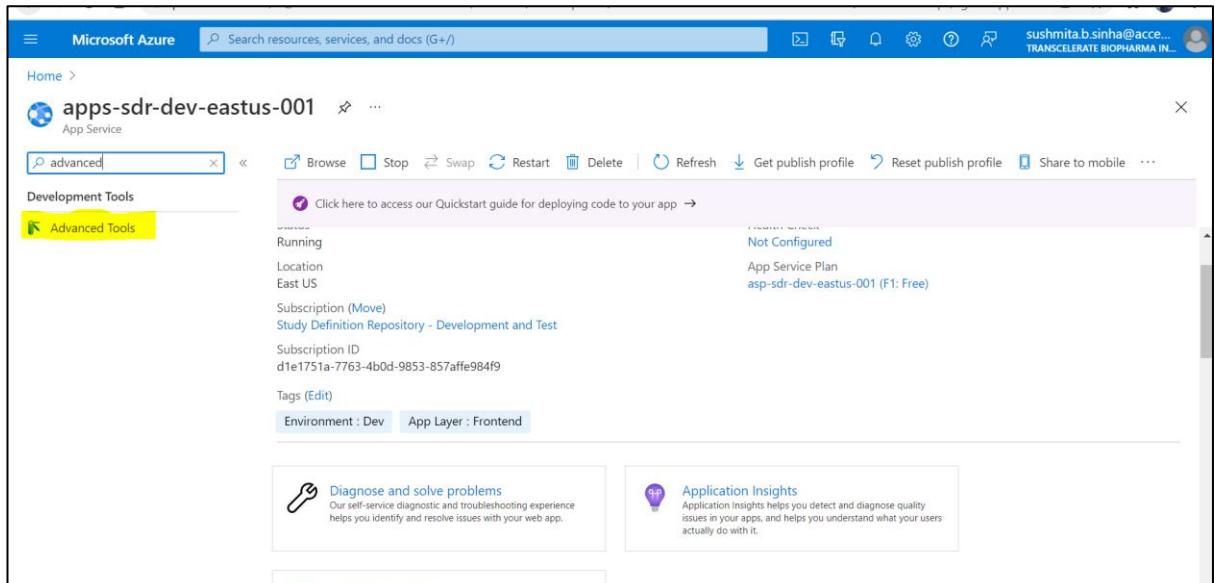
- ii. Select the required resource group and select the UI App Service instance.

Figure 64 Azure Portal - SDR UI App Service



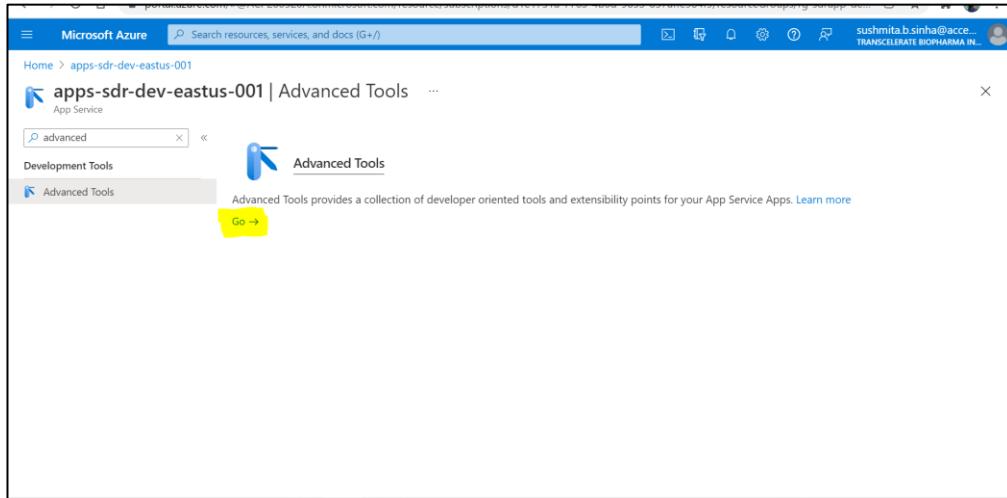
- iii. In search box, search for Advanced tools.

Figure 65 SDR UI App Service - Advanced Tools



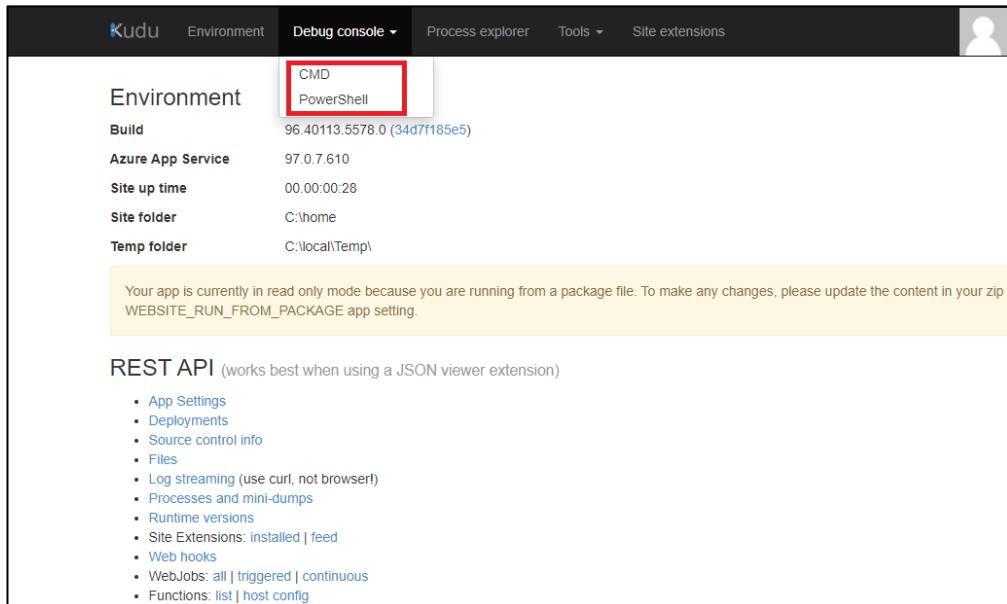
- iv. Click on Go.

Figure 66 SDR UI App Service Advanced Tools - Go



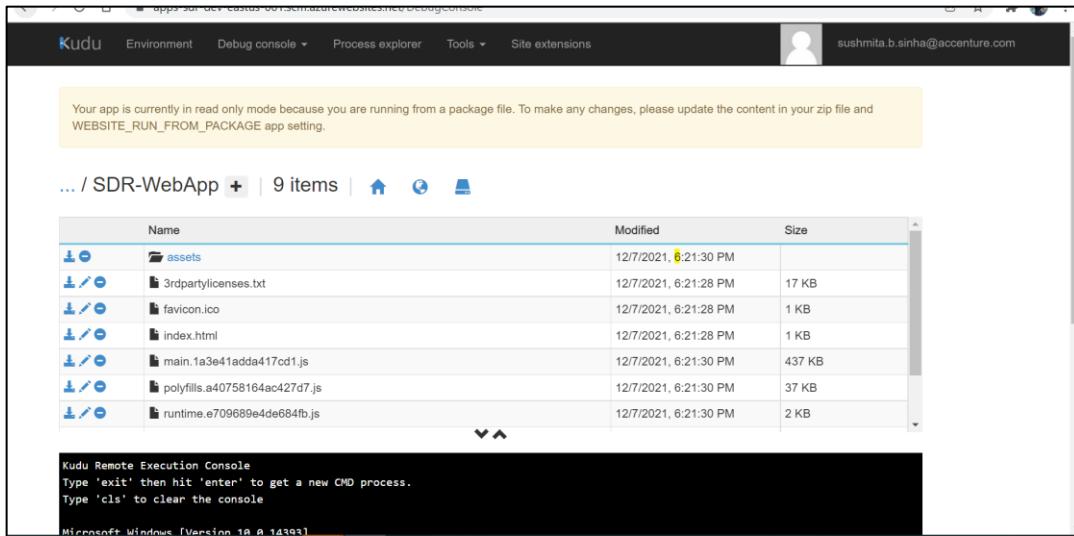
v. Go to Debug console and select CMD/Power Shell

Figure 67 Azure Kudu Tool



vi. Go to, Site -> wwwroot -> Check that the latest code files are deployed.

Figure 68 SDR UI Deployed Files



FOR SDR API BACK-END 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.

5. PaaS Setup

5.1. PaaS Setup for APIM

GOAL:

Configure API Endpoints on API Management (APIM)

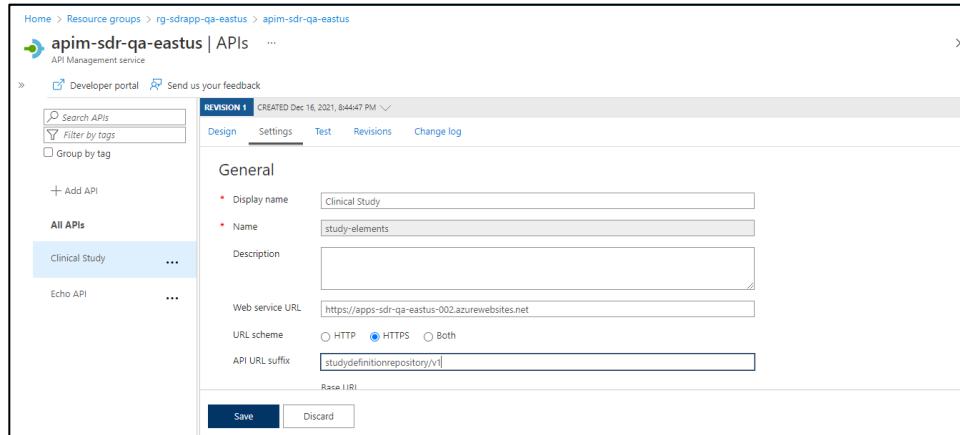
PRE-REQUISITES:

- Contributor access at Resource Group level.

STEPS:

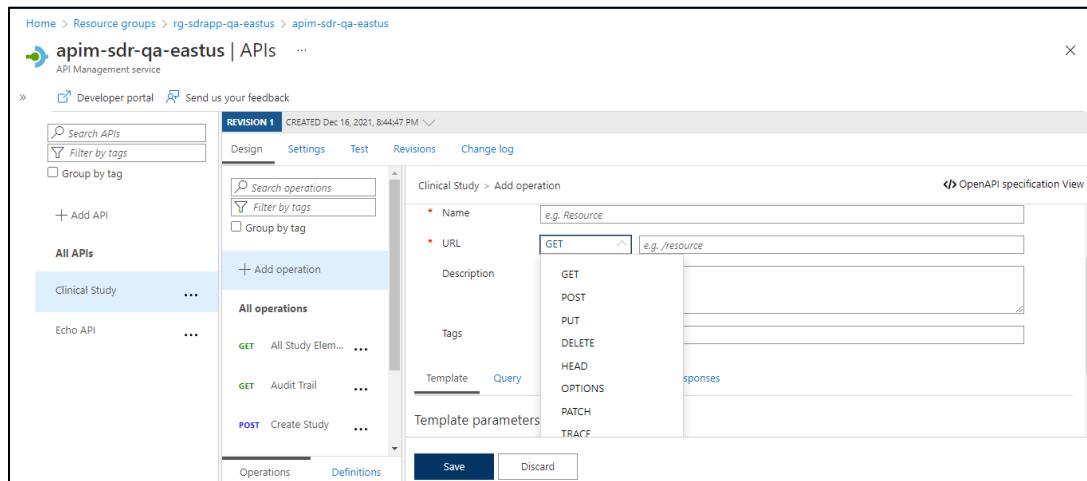
- Go to API Management → Select API Blade → Select the Clinical Study → Set API values by going to the Settings tab

Figure 69 API Management - Name HTTP API



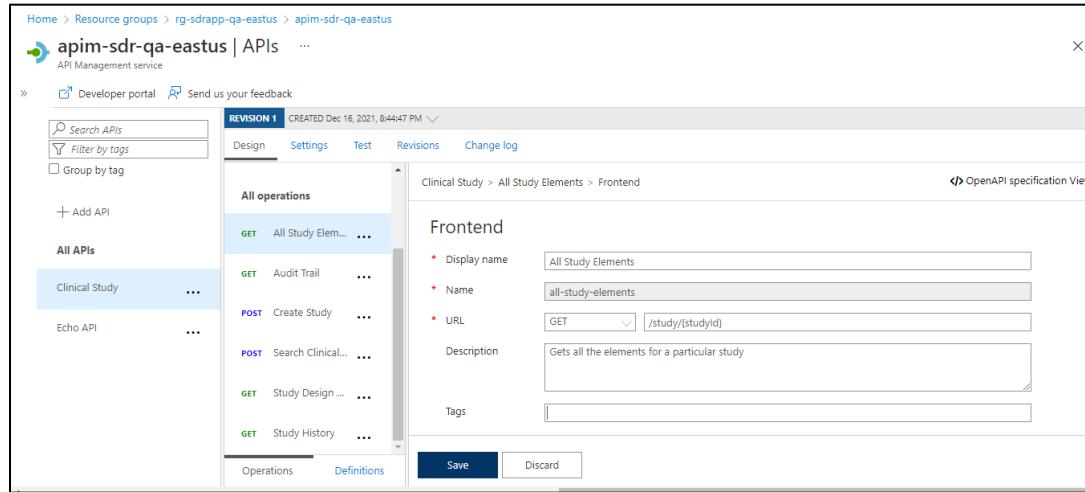
- Display name – Clinical Study.
 - Name – study-elements.
 - Web service URL – <https://apps-sdr-qa-eastus-002.azurewebsites.net> (Backend App Service URL)
 - URL scheme – HTTPS
 - API URL suffix - sdr
 - Products – Unlimited
 - Gateways – Managed
 - Subscription – Uncheck the box
 - Security – None
- ii. Go to design tab and select an operation for the API.

Figure 70 API Management - HTTP API Operation



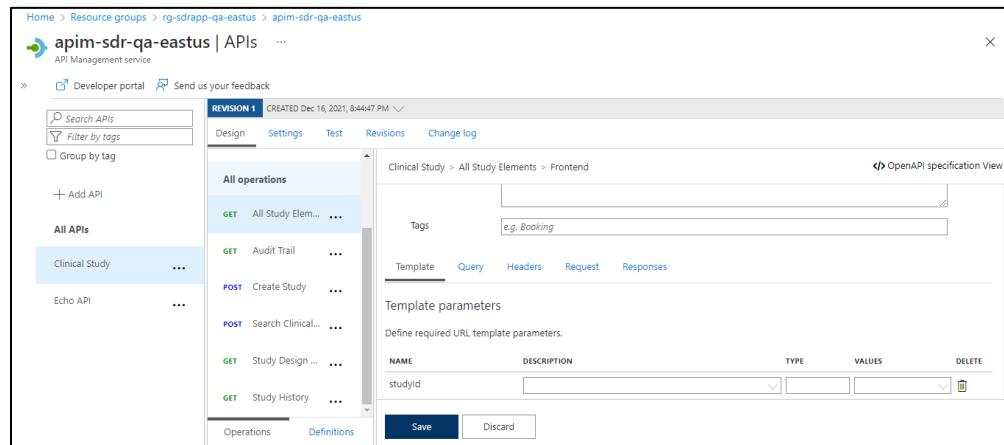
- iii. Fill in the required details of the design template.

Figure 71 API Management - API Operation Details



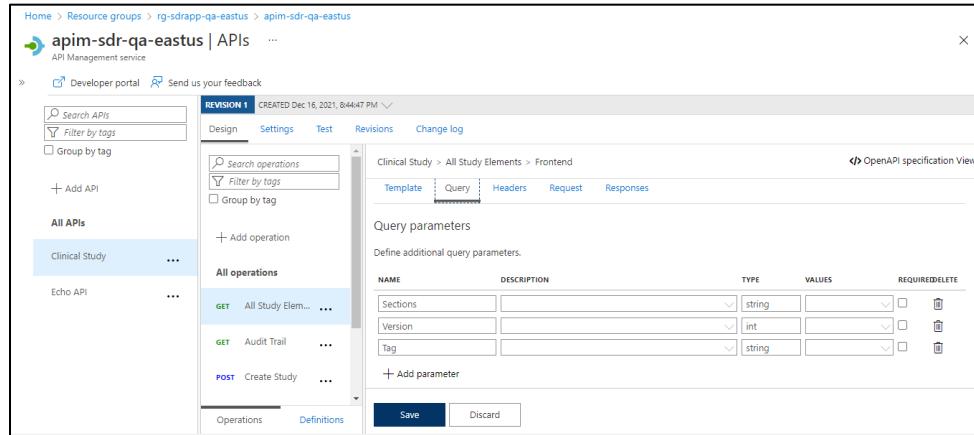
- Display name – All Study Elements
 - Name – It is auto populated based on display name given.
 - URL – GET /study/{studyId}
- iv. Click on template > add parameter – study

Figure 72 API Management – API Operation - Add Parameter



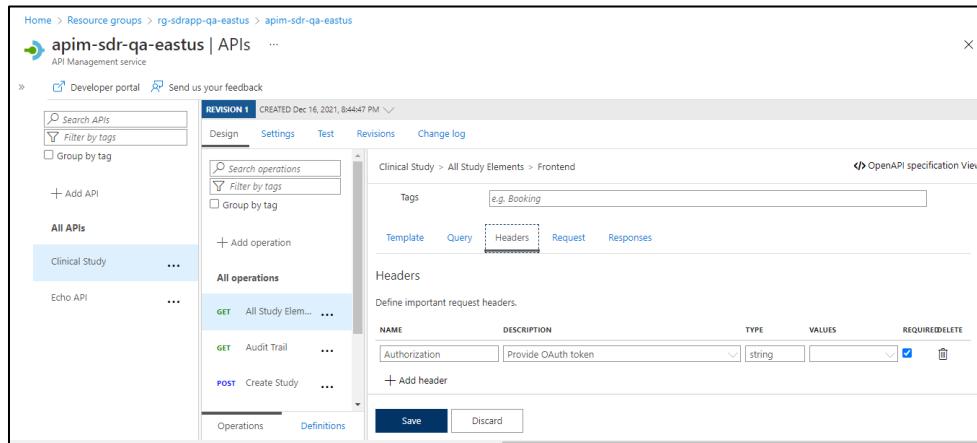
- v. Click on Query > Add Query parameters as below.

Figure 73 API Management – API Operation - Add Query Parameter



- vi. Click on Headers > Add header – Authorization > tick Required checkbox > click save

Figure 74 API Management – API Operation - Add Headers



The below API Operations must be configured by following the steps outlined above for the all the endpoints in SDR API. Refer the DDF_SDR_API User Guide for details of endpoints.

Display Name	URL	Template	Query		Headers	
			Name	Type	Name	Type
All Study Elements	GET /study/{studyId}	studyId	Section s	strin g	Authorizati on	strin g
			Version	int		
			Tag	strin g		
Audit Trail	GET /audittrail/{studyId}	studyId	fromDa te	strin g	Authorizati on	strin g
			toDate	strin g		
Study Design Sections	GET /{studyId}/studydesign/{studydesigndId}	studyId	Section s	strin g	Authorizati on	strin g
			studydesig nId	Version	int	
			Tag	strin g		
Study History	GET /studyhistory		fromDa te	date	Authorizati on	strin g
			toDate	date		
Create Study	POST /study				Authorizati on	strin g
Search Clinical Study	POST /search				Authorizati on	strin g

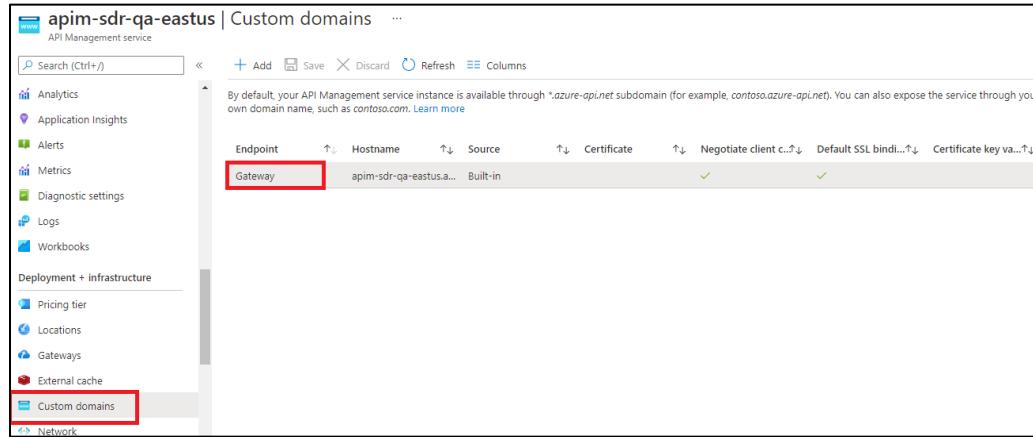
5.2. Secure APIs using client certificate authentication in API

Management

API Management provides the capability to secure access to APIs (i.e., client to API Management) using client certificates. You can validate certificates presented by the connecting client and check certificate properties against desired values using policy expressions.

To receive and verify client certificates over HTTP/2 turn on the "Negotiate client certificate" setting on the "Custom domains" blade as shown below.

Figure 75 API Management - Custom Domains

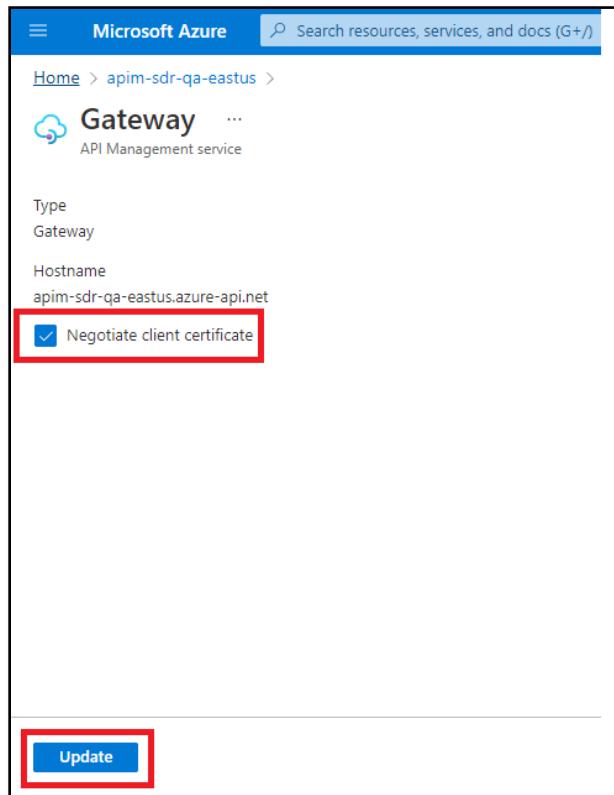


The screenshot shows the 'Custom domains' blade for the API Management service 'apim-sdr-qa-eastus'. The table lists one entry:

Endpoint	Hostname	Source	Certificate	Negotiate client certificate	Default SSL binding	Certificate key value
Gateway	apim-sdr-qa-eastus.azure-api.net	Built-in		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

The left sidebar shows navigation links for Analytics, Application Insights, Alerts, Metrics, Diagnostic settings, Logs, Workbooks, Deployment + infrastructure (Pricing tier, Locations, Gateways, External cache), and Custom domains. The 'Custom domains' link is highlighted with a red box.

Figure 76 API Management - Custom Domains - Gateway



The screenshot shows the 'Gateway' configuration page for the API Management service 'apim-sdr-qa-eastus'. The 'Type' is set to 'Gateway'. The 'Hostname' is 'apim-sdr-qa-eastus.azure-api.net'. The 'Negotiate client certificate' checkbox is checked and highlighted with a red box. At the bottom, there is a large blue 'Update' button highlighted with a red box.

CREATE ROOT CA CERTIFICATE:

- i. Run the below commands in Power Shell in local system (Run as Administrator) to create the Root CA certificate Copy the thumbprint value generated to use during Client Certificate creation.

```
$rootcert = New-SelfSignedCertificate -CertStoreLocation cert:\CurrentUser\My -DnsName "SDR APIM" -KeyUsage CertSign
```

```
Write-host "Certificate Thumbprint: $($rootcert.Thumbprint)"
```

```
Export-Certificate -Cert $rootcert -FilePath C:\certtemp\SCDudesRootCA.cer
```

Directory: C:\certtemp

Mode	LastWriteTime	Length	Name
-a---	12/14/2021 1:17 PM	847	TranscelerateRootCA.cer

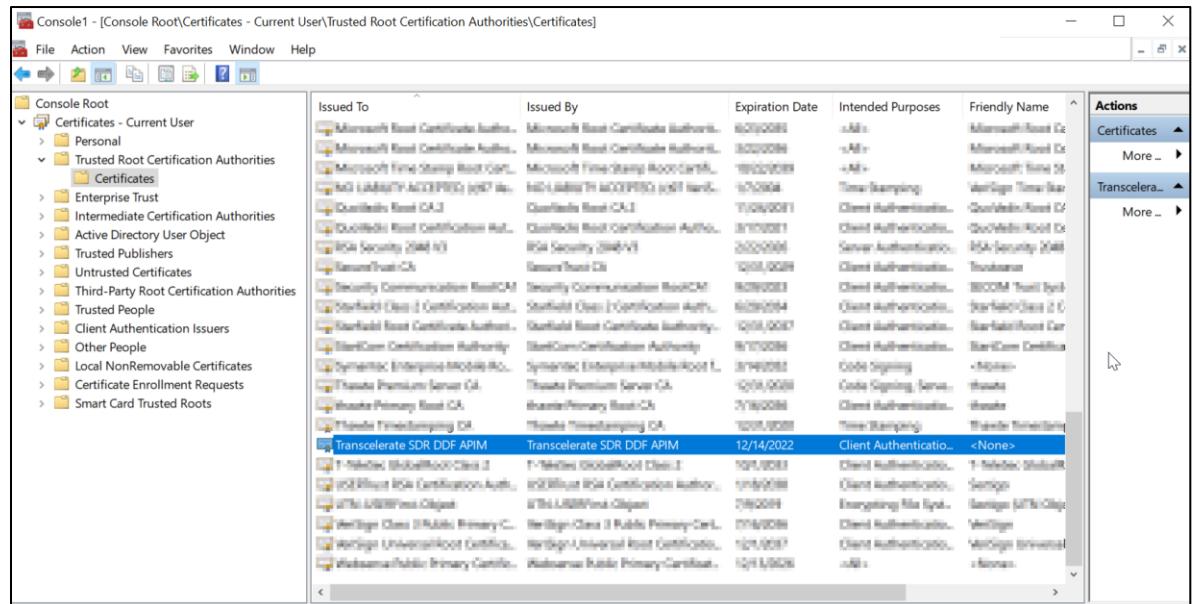
- ii. Importing the certificate to trusted publishers on local system

#Imports certificate to Trusted Publishers

```
Import-Certificate -FilePath C:\certtemp\SCDudesRootCA.cer -CertStoreLocation Cert:\LocalMachine\Root
```

iii. Now the Root CA certificate is available to access under your local system MMC snap-in where you can view the metadata of the certificate

Figure 77 Root CA Certificate in MMC



CREATE CLIENT CERTIFICATE:

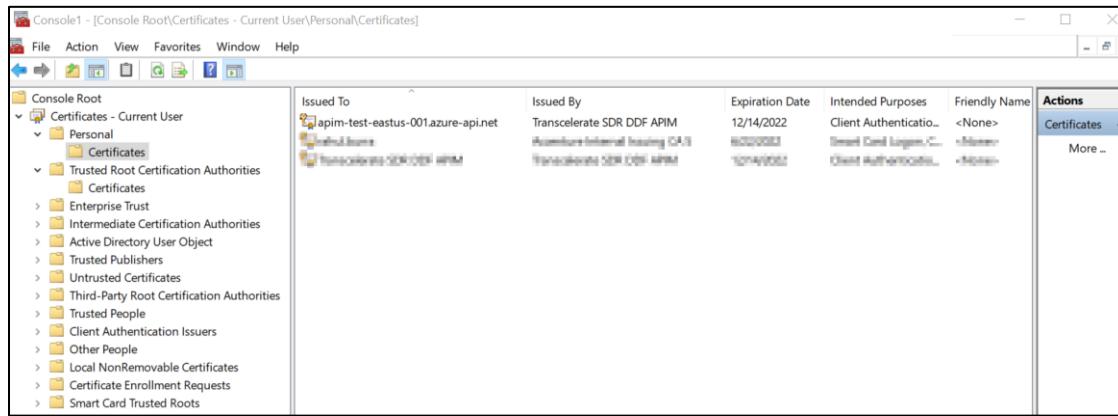
- i. Run the below commands to create the Client certificate based on the Root CA certificate. The thumbprint in this command is of the Root CA certificate.

```
$rootca = Get-ChildItem cert:\CurrentUser\my | Where-Object
{$_._Thumbprint -eq "3C680962BB07CBEDD787C657A88DD74671044180"}

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

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

Figure 78 Client Certificate

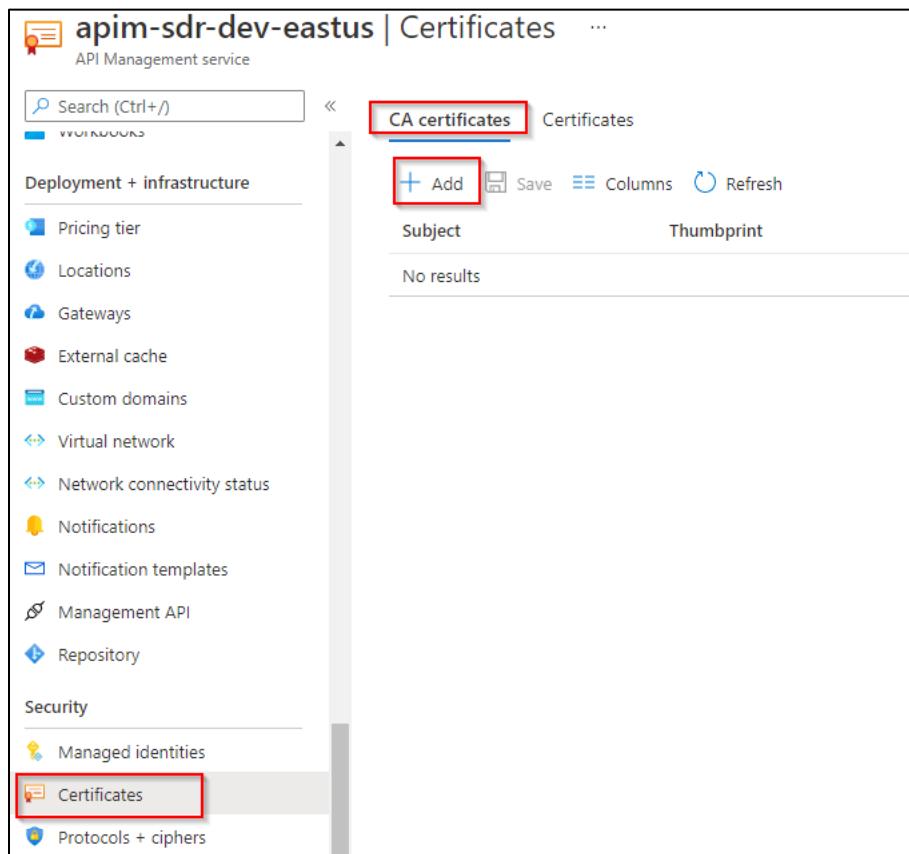


- iii. Export the certificate in .pfx format and set password when prompted.

UPLOAD THE CLIENT CERTIFICATE TO APIM:

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

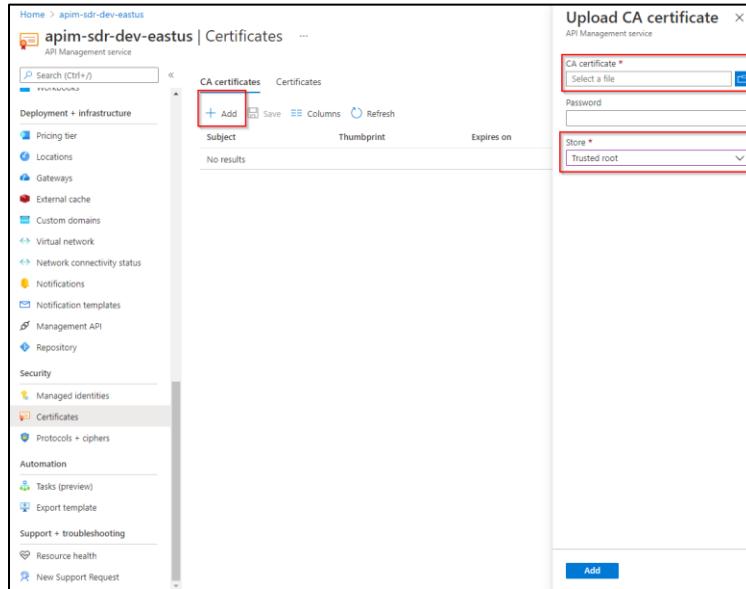
Figure 79 APIM Certificates



The screenshot shows the 'Certificates' blade for the 'apim-sdr-dev-eastus' API Management service. The 'CA certificates' tab is active. At the top right, there is an 'Add' button, which is highlighted with a red box. Below it are 'Save', 'Columns', and 'Refresh' buttons. The main area displays two columns: 'Subject' and 'Thumbprint'. A message 'No results' is shown. On the left side, there is a navigation menu with several sections: 'Deployment + infrastructure' (including Pricing tier, Locations, Gateways, External cache, Custom domains, Virtual network, Network connectivity status, Notifications, Notification templates, Management API, Repository), 'Security' (including Managed identities, Certificates, and Protocols + ciphers), and 'APIs' (with a 'Search (Ctrl+/' input field). The 'Certificates' link in the 'Security' section is also highlighted with a red box.

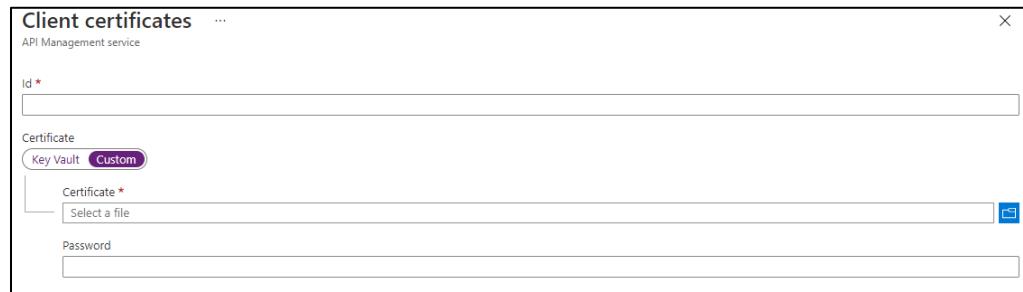
- ii. After clicking on the Add option upload the Root CA certificate (.cer) file to the “Trusted root” by selecting the root CA file as below.
Note: Refer

Figure 80 APIM Certificates - Add CA Certificate



- iii. Go to Certificates option and click on “Add” option and upload the password protected Client certificate (.pfx) format as shown below.

Figure 81 APIM Certificates - Client Certificates



- iv. Once both the certificates are uploaded it should be visible on the API Management certificates blade as below

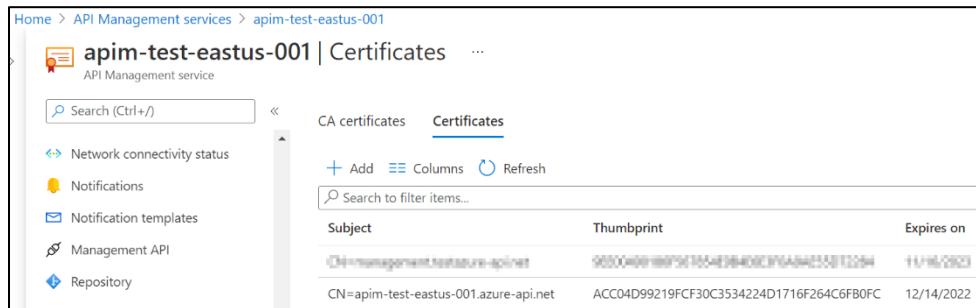
Root CA Certificate

Figure 82 APIM Certificates - Root CA



Client Certificate

Figure 83 APIM Certificates - Client Certificate

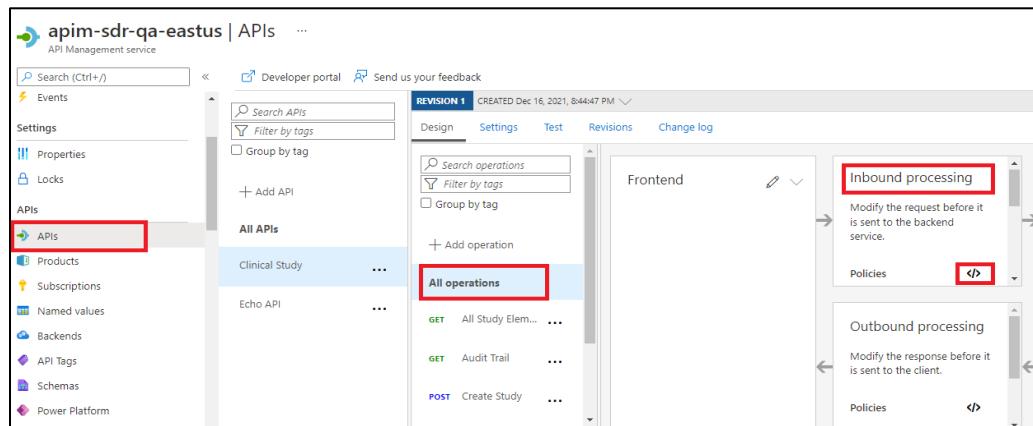


The screenshot shows the 'Certificates' tab selected in the APIM instance 'apim-test-eastus-001'. It displays a table with columns for Subject, Thumbprint, and Expires on. There is one entry listed:

Subject	Thumbprint	Expires on
CN=management-test.eastus.apim.net	9E5004981B8F5C7854E9B4083F5A9A25872294	11/16/2023
CN=apim-test-eastus-001.azure-api.net	ACC04D99219FCF30C3534224D1716F264C6FB0FC	12/14/2022

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

Figure 84 APIM Inbound Processing



The screenshot shows the 'Inbound processing' section for the 'Clinical Study' API operation. A red box highlights the 'Inbound processing' section, which contains a policy code snippet:

```

<inbound>
    <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>
</inbound>

```

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

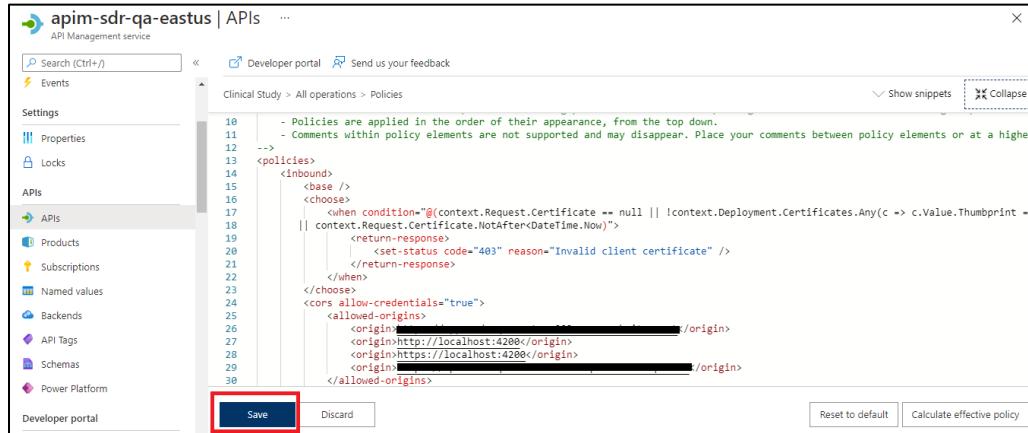
```

<policies>
    <inbound>
        <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>
    </inbound>

```

```
        </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>
    </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>
```

Figure 85 APIM - Inbound Policy



The screenshot shows the Azure API Management service interface for an API named "apim-sdr-qa-eastus". The left sidebar lists various settings like Properties, Locks, APIs, Products, Subscriptions, Named values, Backends, API Tags, Schemas, and Power Platform. The "APIs" section is selected. The main area displays an XML-based inbound policy configuration:

```

10  - Policies are applied in the order of their appearance, from the top down.
11  . Comments within policy elements are not supported and may disappear. Place your comments between policy elements or at a higher
12  -->
13  <policies>
14  <inbound>
15  <base />
16  <choose>
17  <when condition="@((context.Request.Certificate == null || !context.Deployment.Certificates.Any(c => c.Value.Thumbprint =="
18  || context.Request.Certificate.NotAfter<DateTime.Now))">
19  <return-response>
20  <set-status code="403" reason="Invalid client certificate" />
21  </return-response>
22  </when>
23  </choose>
24  <cors allow-credentials="true">
25  <allowed-origins>
26  <origin>[REDACTED]/origin</origin>
27  <origin>http://localhost:4200</origin>
28  <origin>https://localhost:4200</origin>
29  <origin>[REDACTED]</origin>
30  </allowed-origins>

```

The "Save" button at the bottom left is highlighted with a red box.