TransCelerate BIOPHARMA INC.

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

List of Figures

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.

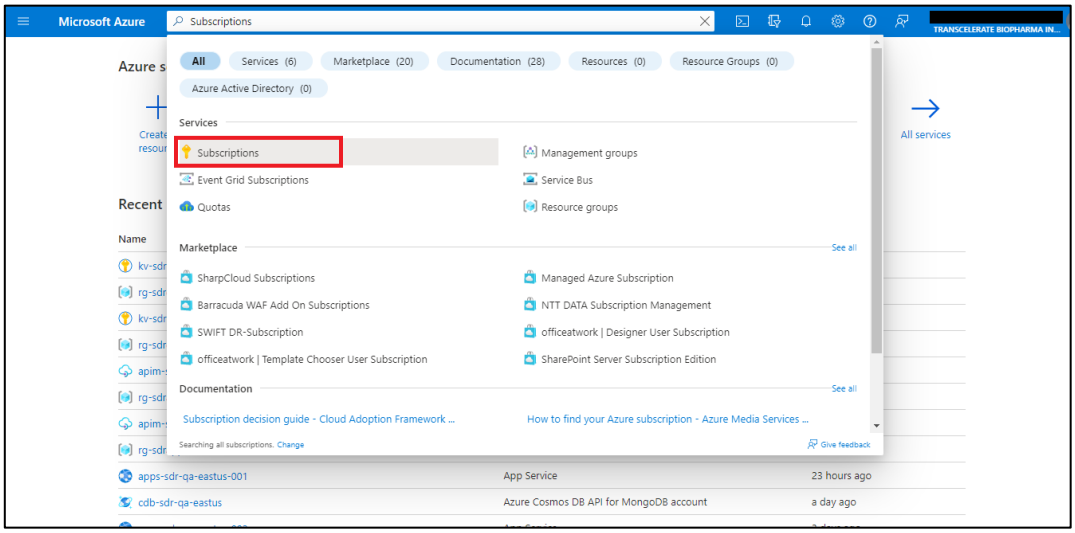
PRE-REQUISITES:

Global Admin or Subscription Owner level of access to Azure.

STEPS:

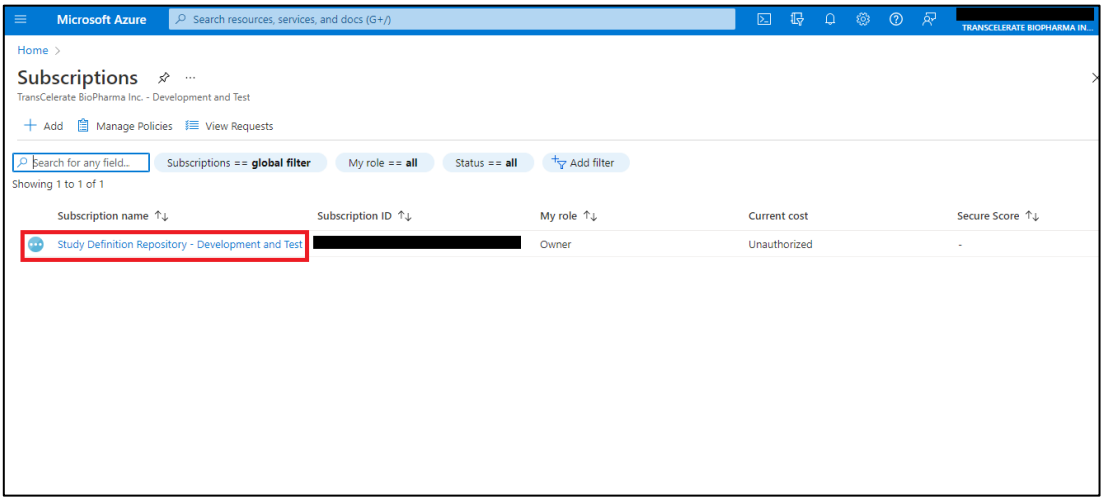
i. Sign into the Azure portal.
ii. On the Azure portal menu, search for Subscriptions. Select it from the available options as shown below.

Figure 1 Select Subscriptions in Azure Portal



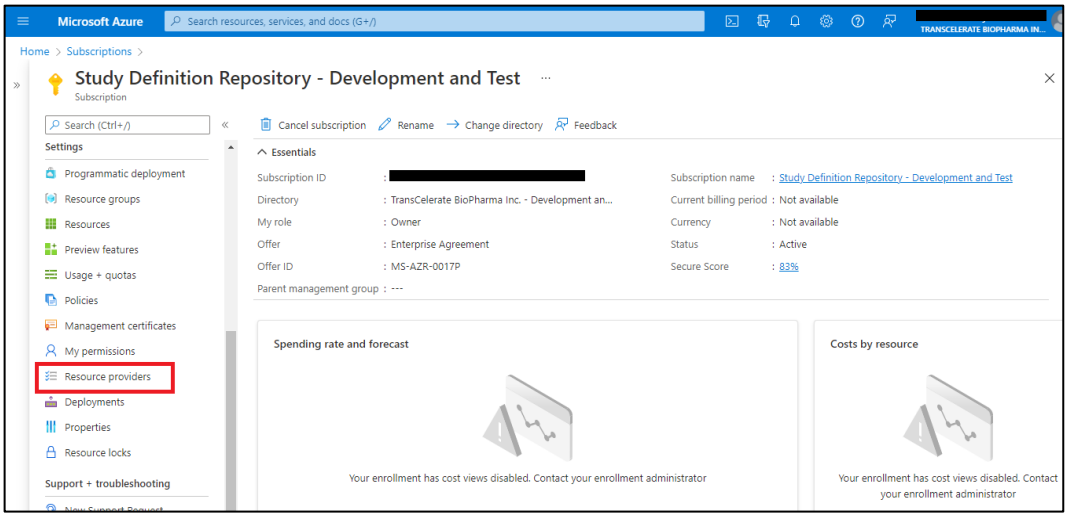
iii. Select the subscription you want to view.

Figure 2 Select Subscription



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

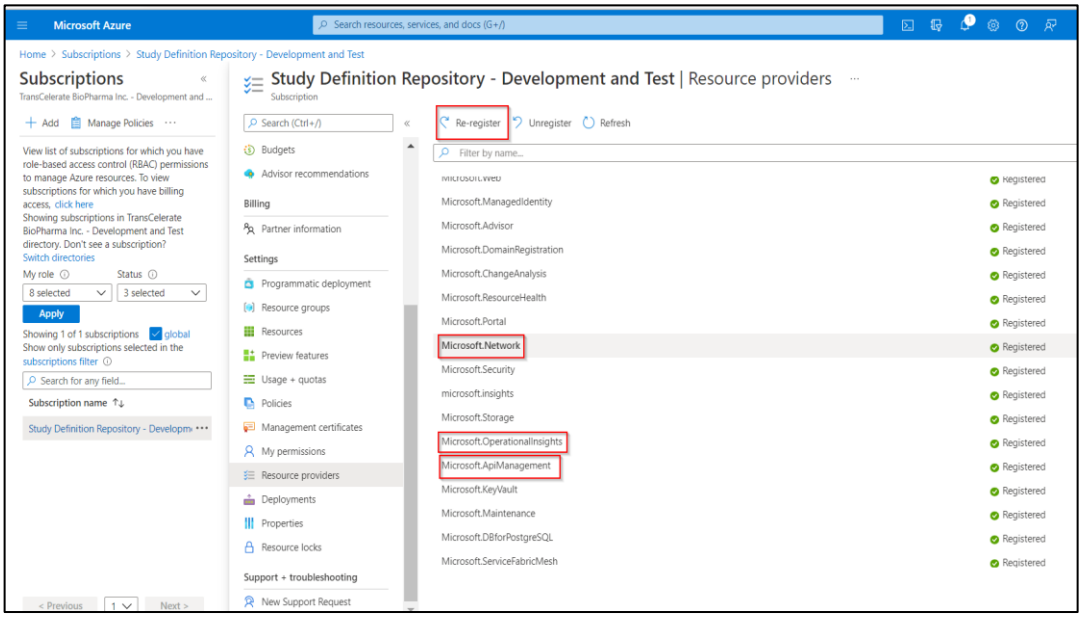
Figure 3 Resource Providers in a Subscription



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.
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

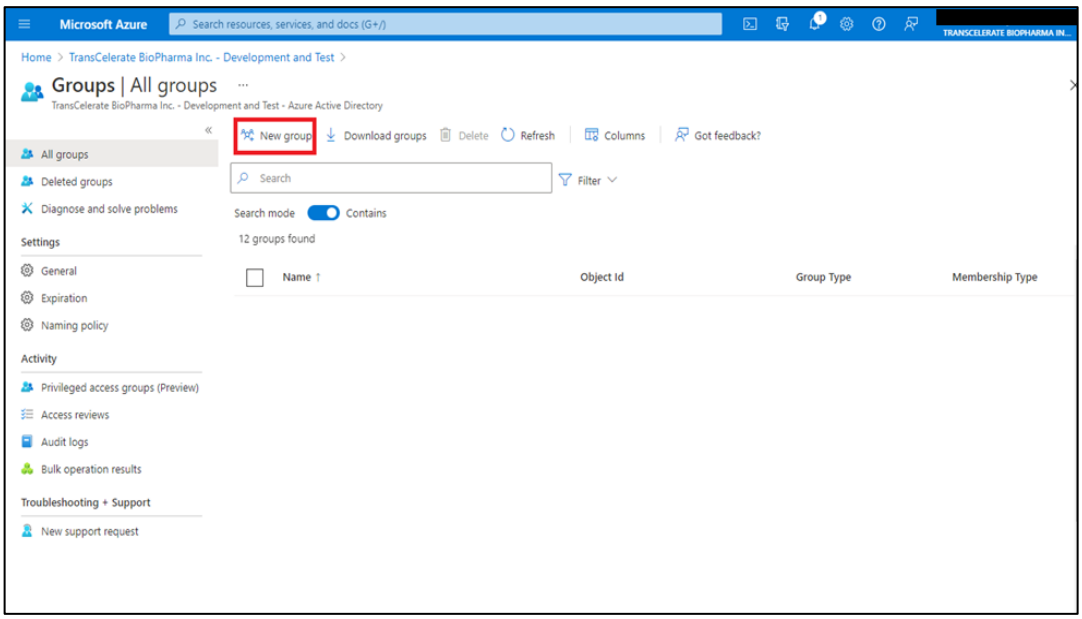
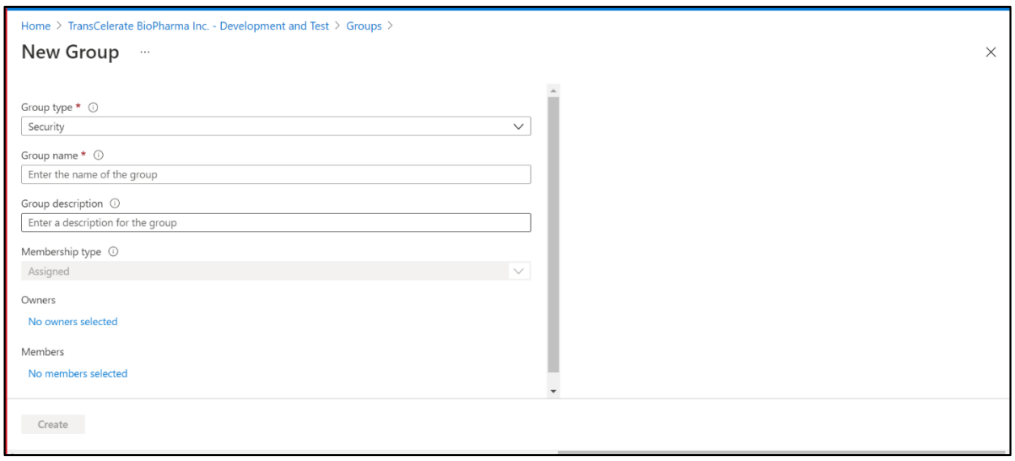


Figure 6 Add New Group



2.3. Create Users

GOAL:

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

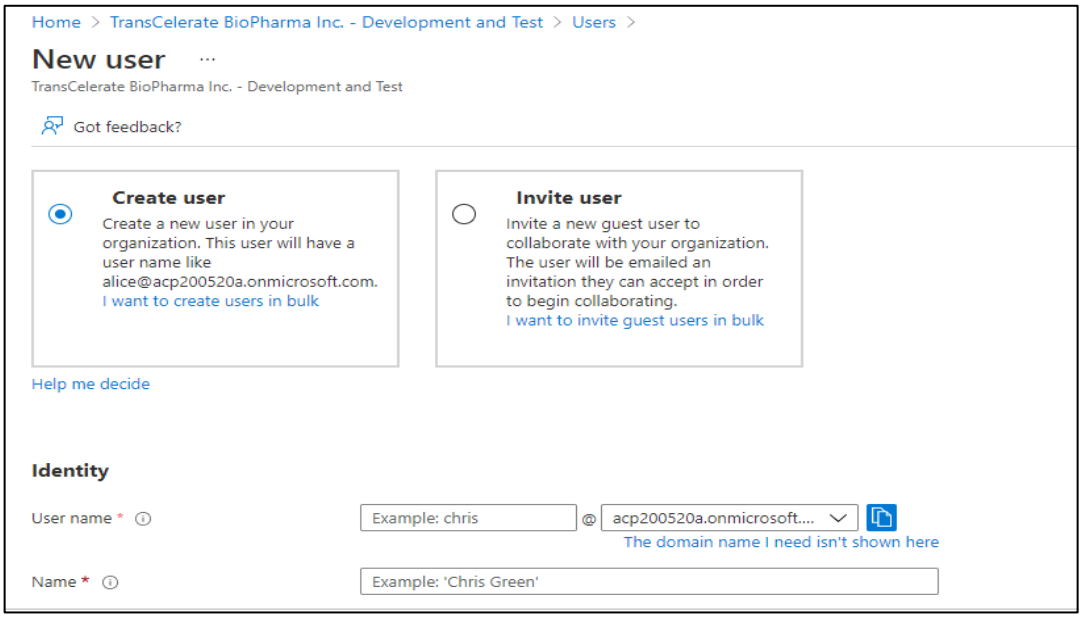
PRE-REQUISITES:

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

STEPS:

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

Figure 7 Create New User



2.4. Role Based Access Controls (RBAC)

GOAL:

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

PRE-REQUISITES:

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

STEPS:

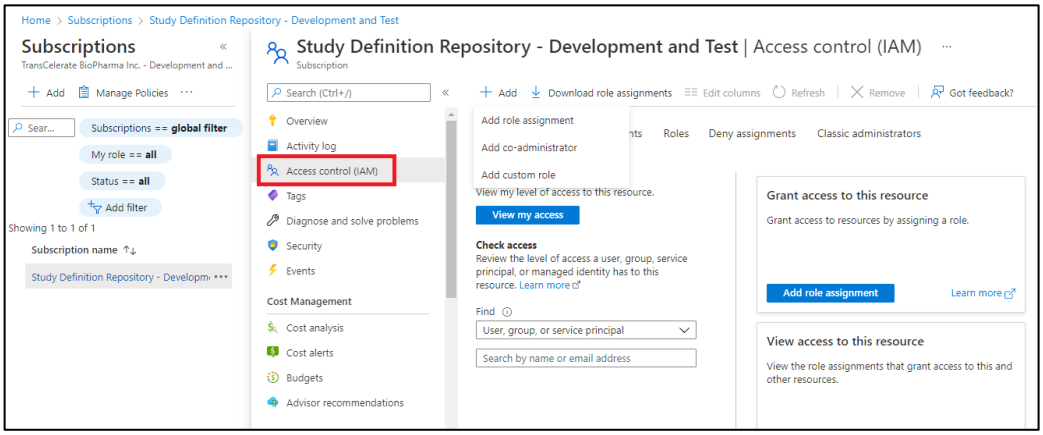
Access Control (IAM) is to limit access and assign roles to groups at the subscription, resource group, and resource level.

At subscription level

i. Go to the subscription.

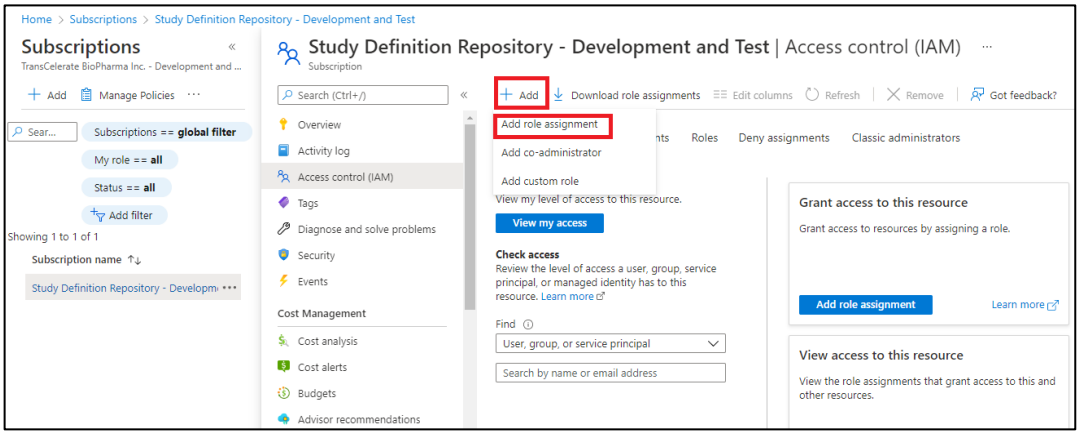
ii. On the left pane select Access control (IAM) as shown in below screenshot.

Figure 8 Access Control (IAM)



iii. Click on + Add and add the role assignment

Figure 9 Add Role Assignment



iv. Select the required role (ex: reader, contributor etc.,) 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

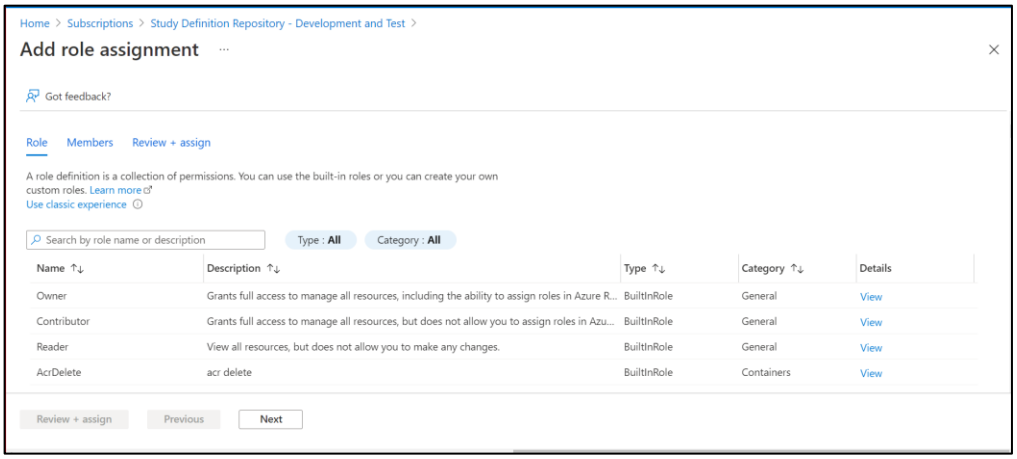
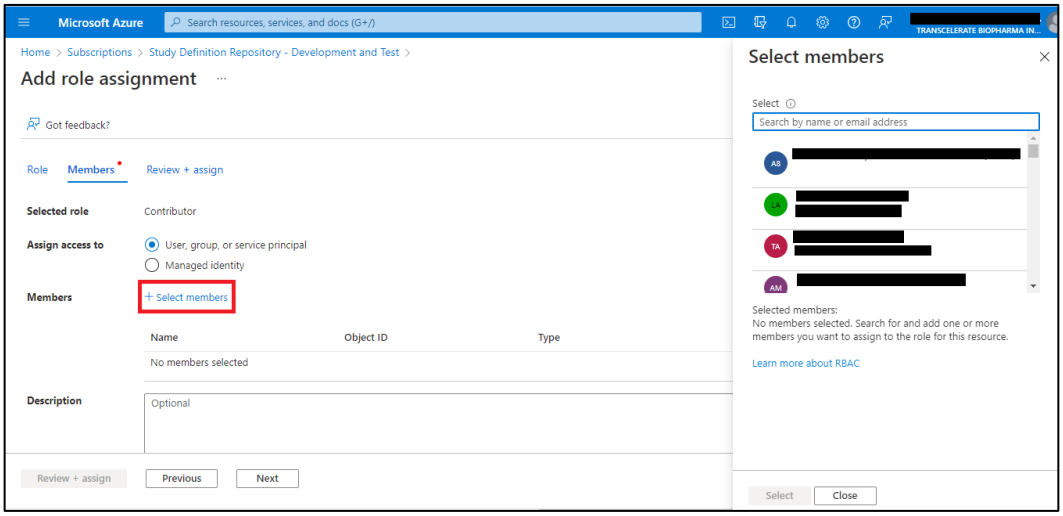
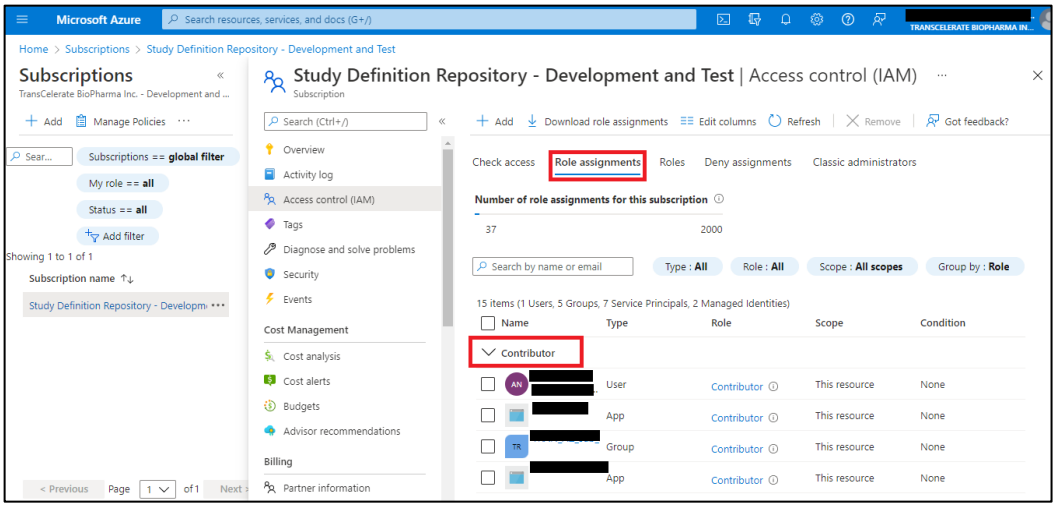


Figure 11 Select Members for Role Assignment



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

Figure 12 View Role Assignments



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.

i. Create Storage Account and Blob Container for storing State file remotely.
ii. 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

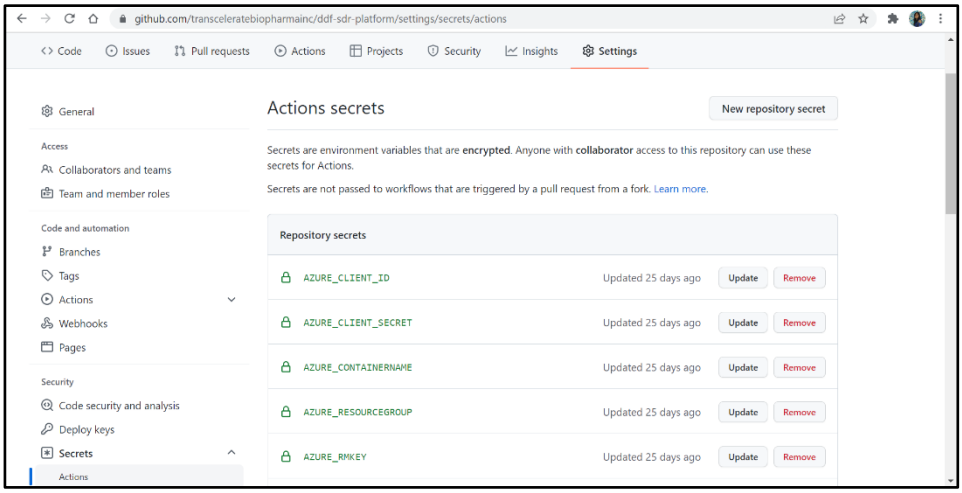
Secret Name	Values
AZURE_SP	The Service Principal details in JSON format.
AZURE_RESOURCEGROUP	The name of the Resource Group that contains the storage account.
AZURE_STORAGEACCOUNT	The Storage Account name
AZURE_CONTAINERNAME	The name of the blob container wherein the Terraform State file will be stored.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

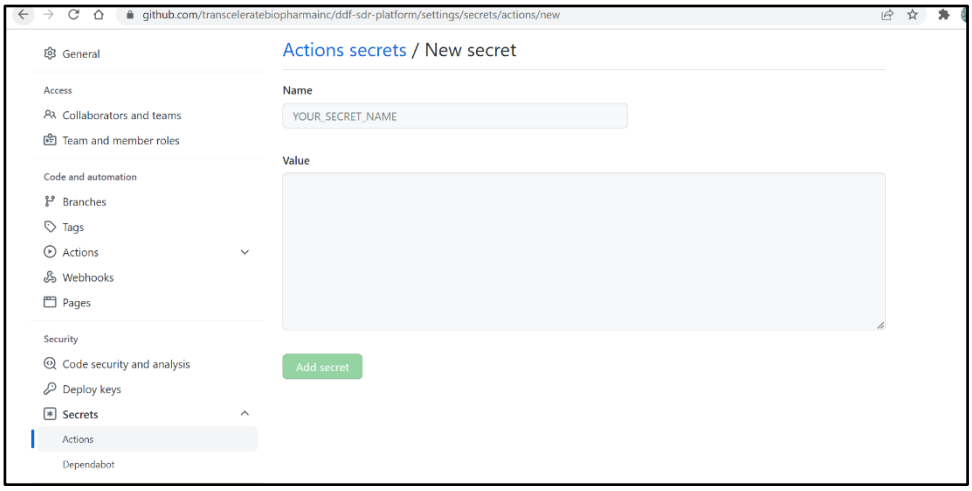
i. Go to repo settings.
ii. On the lower left-hand side of the screen, click on Secrets.
iii. Under that click on Actions.
iv. Then click on New Repository Secret.

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:

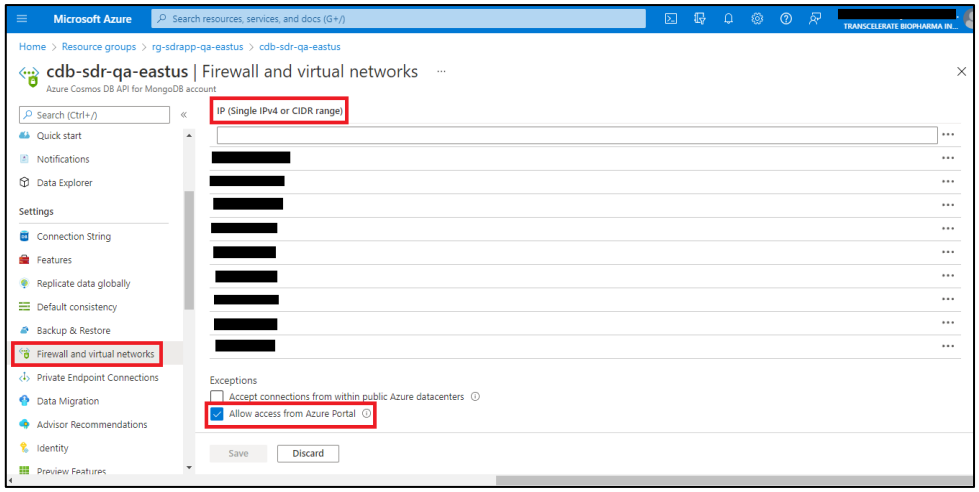
- Contributor level of access for Resource Group.

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

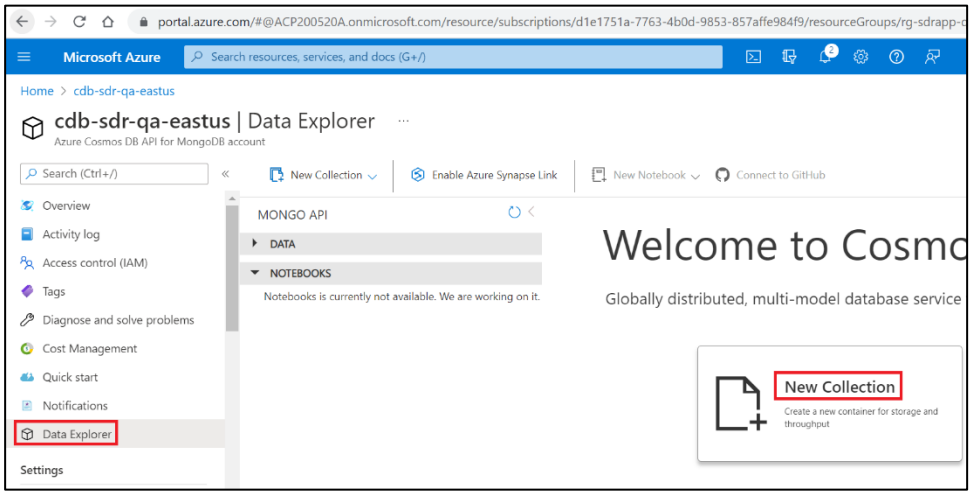
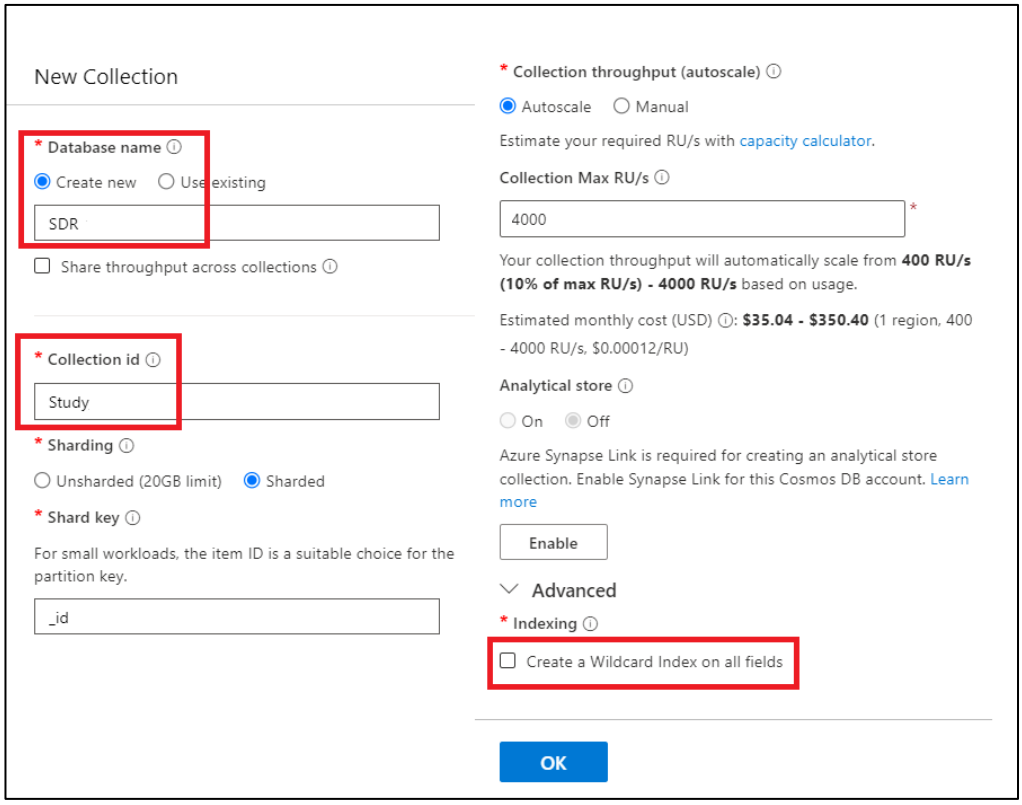
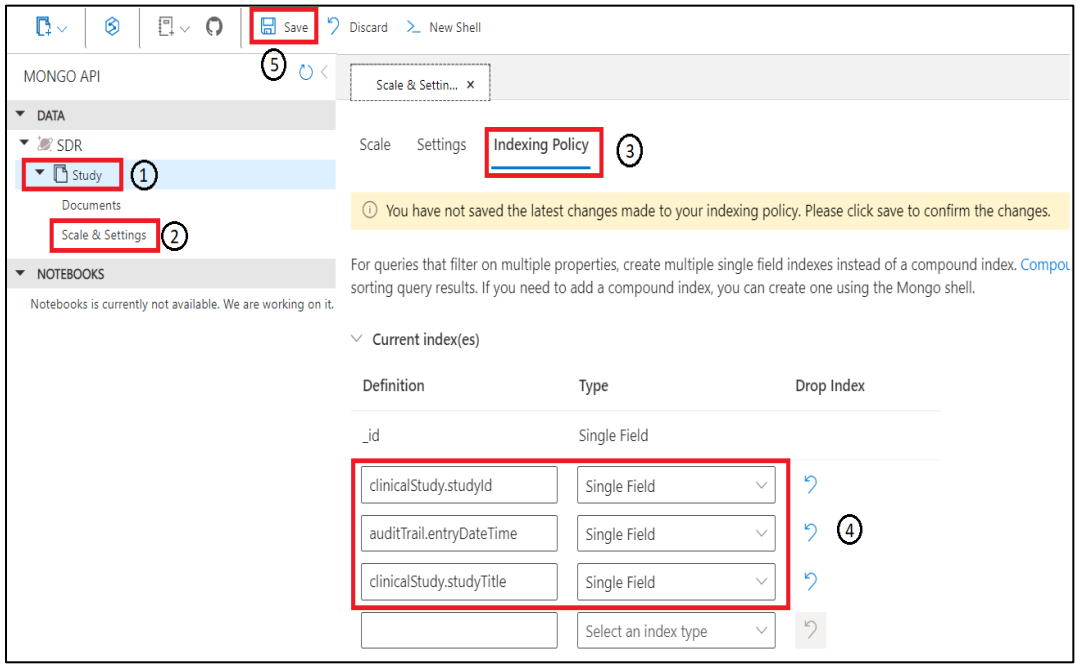


Figure 17 Creating Collection in CosmosDB



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:

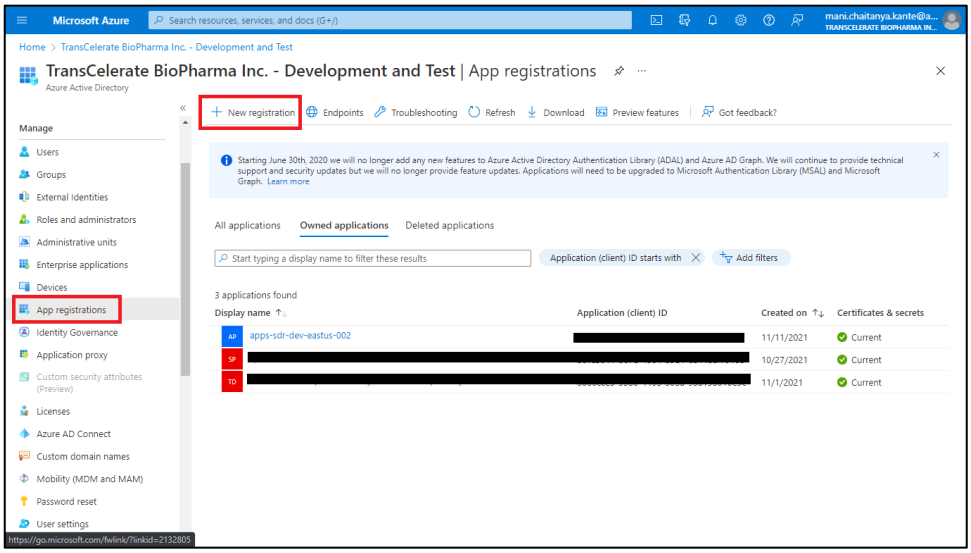
- Global Administrator level of access at Active Directory level.

STEPS

SDR UI APP REGISTRATION:

i. Go To Azure AD → App Registrations → New Registration

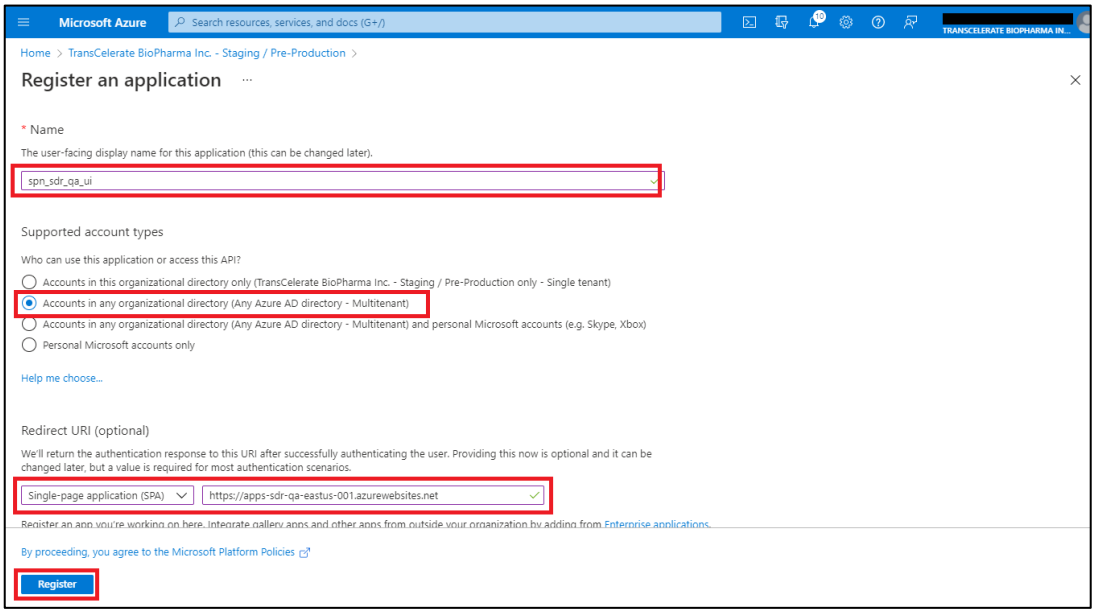
Figure 19 Azure AD - App Registration



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)

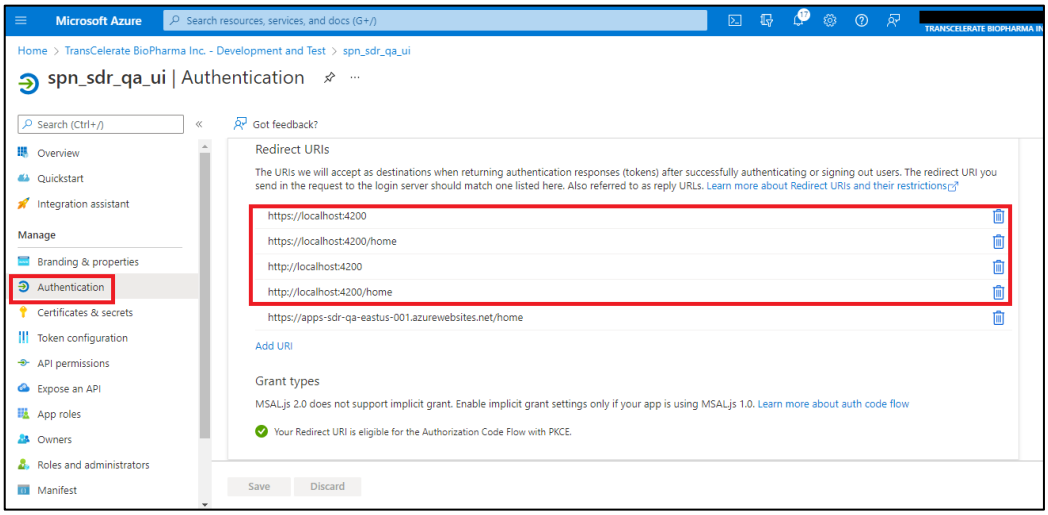
iii. Click on Register to save the App registration.

Figure 20 Azure AD - Register Application



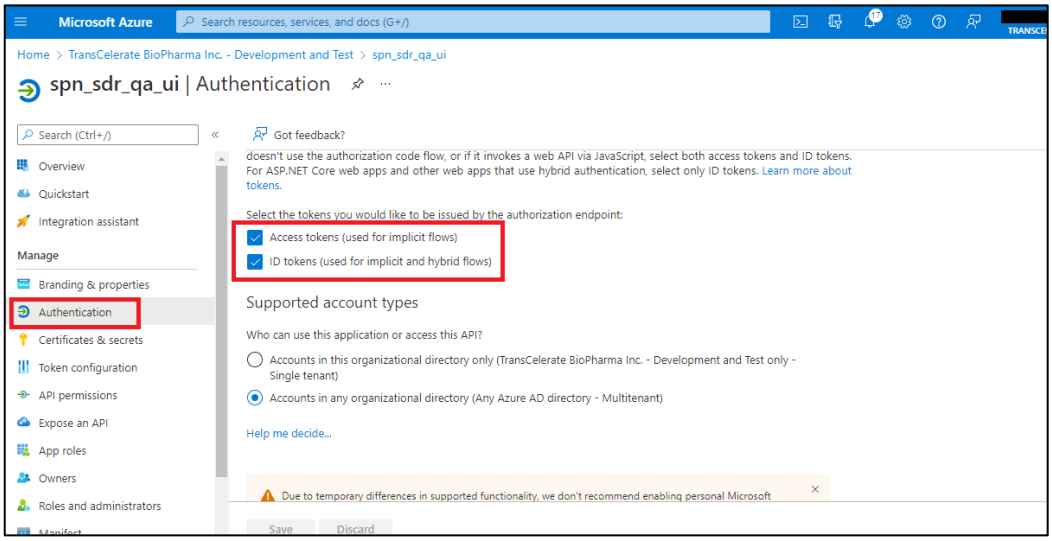
iv. Go to Authentication blade, add additional Redirect URL as needed. Add localhost URL's for testing the application from development machine.

Figure 21 Azure AD - App Registration - Redirect URI



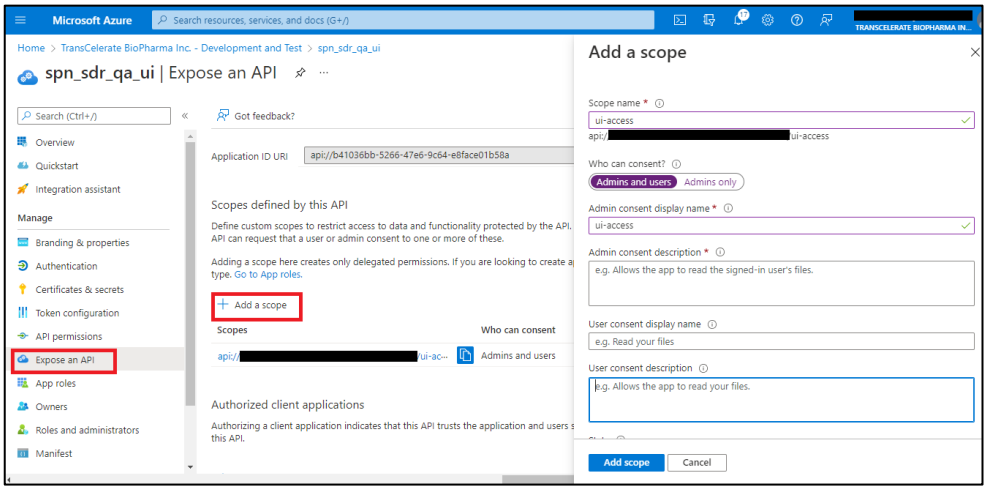
v. Check to select “Access tokens” and “ID tokens”

Figure 22 Azure AD - App Registration - Authentication



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

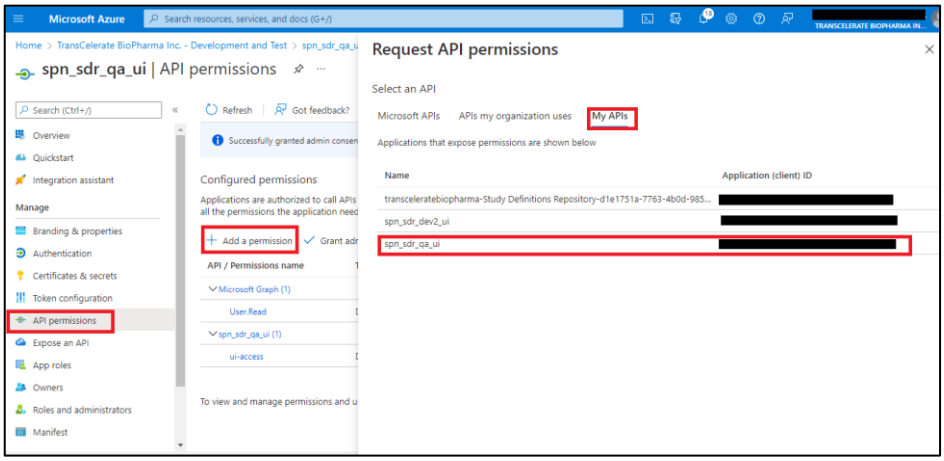
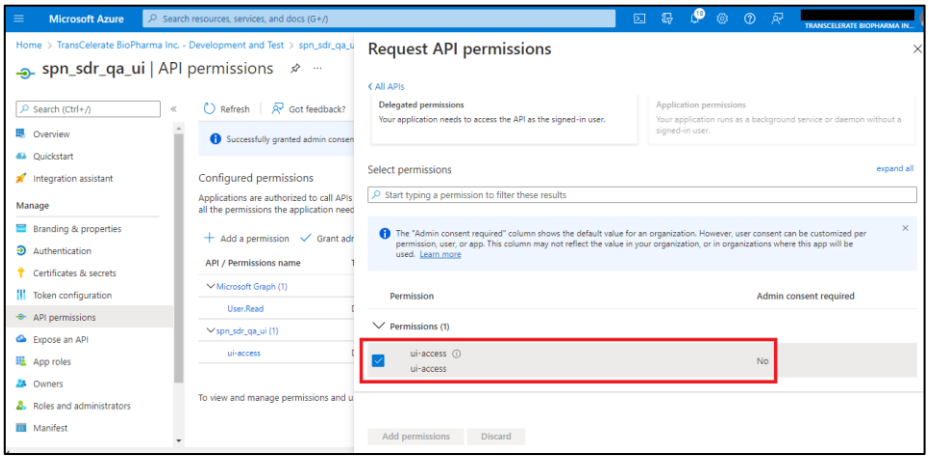
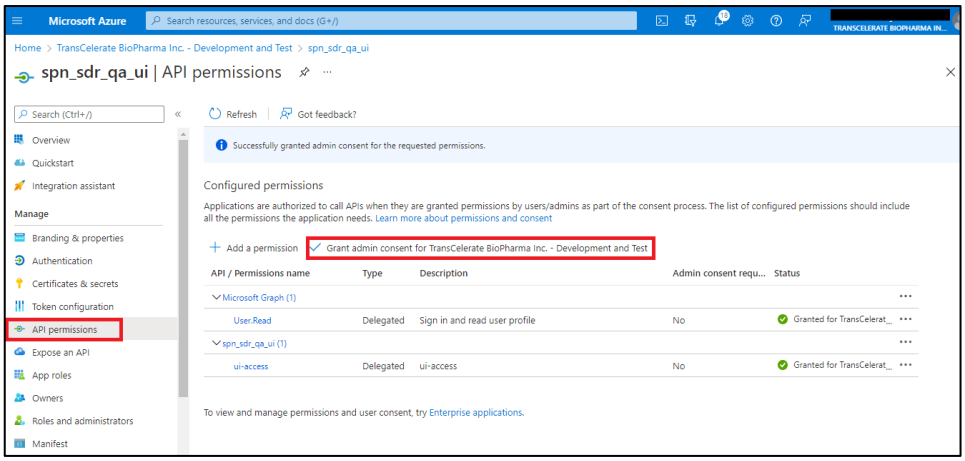


Figure 25 Azure AD - App Registration - Add API Permission



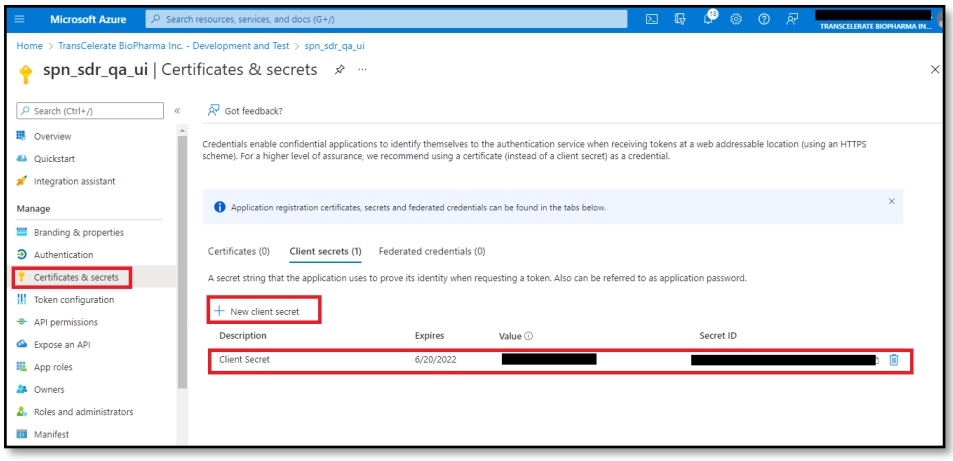
viii. Grant Admin consent.

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



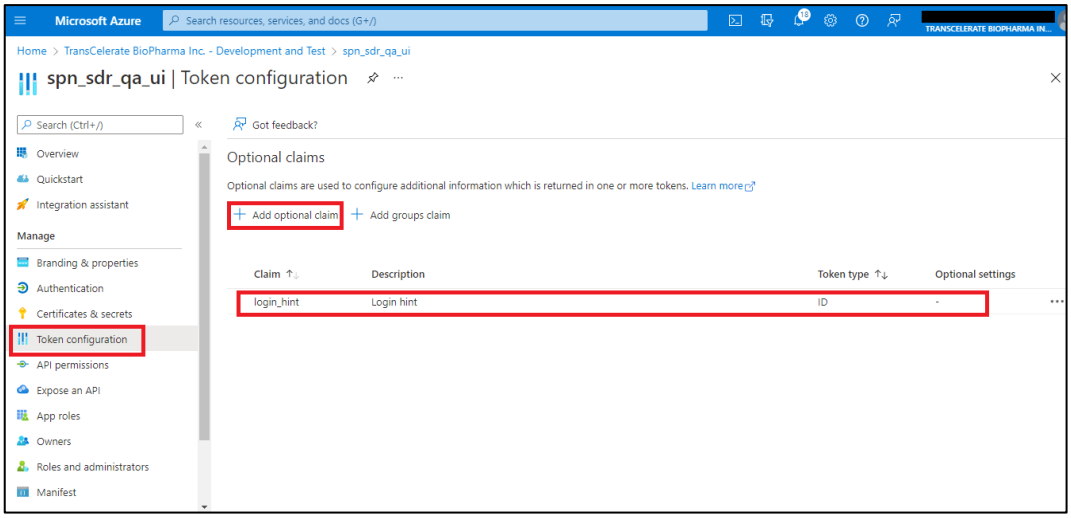
ix. Go to certificates & secrets → click on client secrets → click new client secret and copy the value and add it on Key Vault secrets.

Figure 27 Azure AD - App Registration - Certificates & Secrets



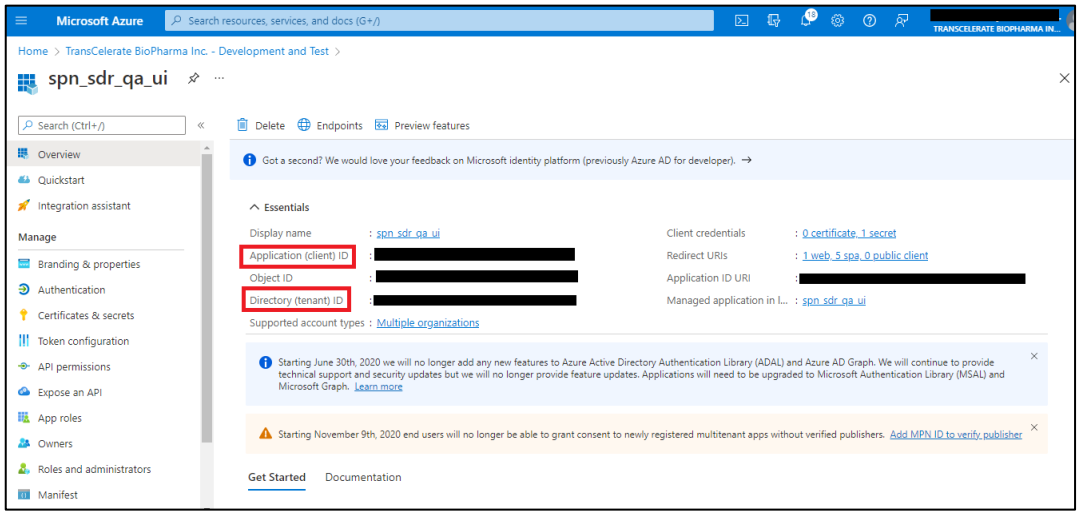
x. Go to Token Configuration and click “Add optional Claim” to add “Login_hint”.

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:

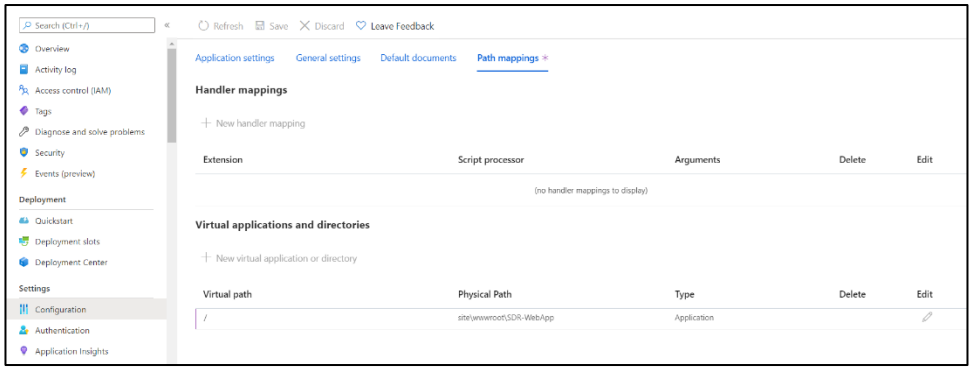
Update the Path Mapping of UI Azure App Service.

PRE-REQUISITES:

- Contributor access at Resource Group level.

STEPS:

i. Navigate to UI App Service instance in App resource group.
ii. Go to Configuration → Path Mappings
iii. 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.

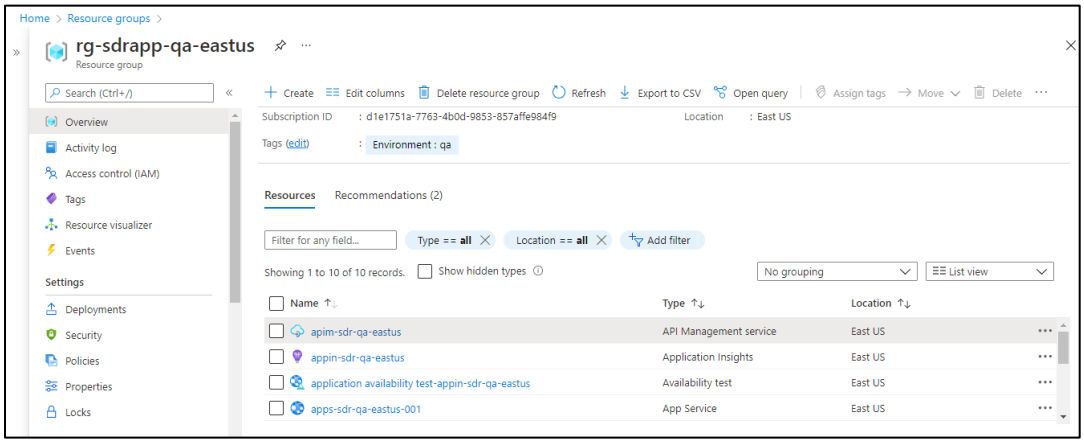
PRE-REQUISITES:

- Contributor access at Resource Group level.

STEPS:

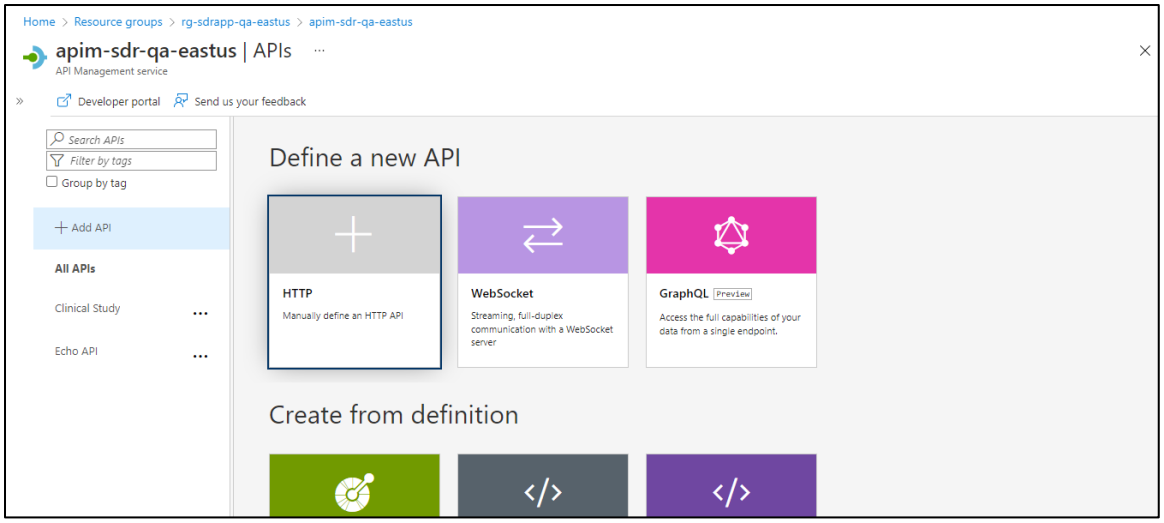
iv. Navigate to APIM instance in App resource group.

Figure 30 Azure Portal - SDR App Resource Group



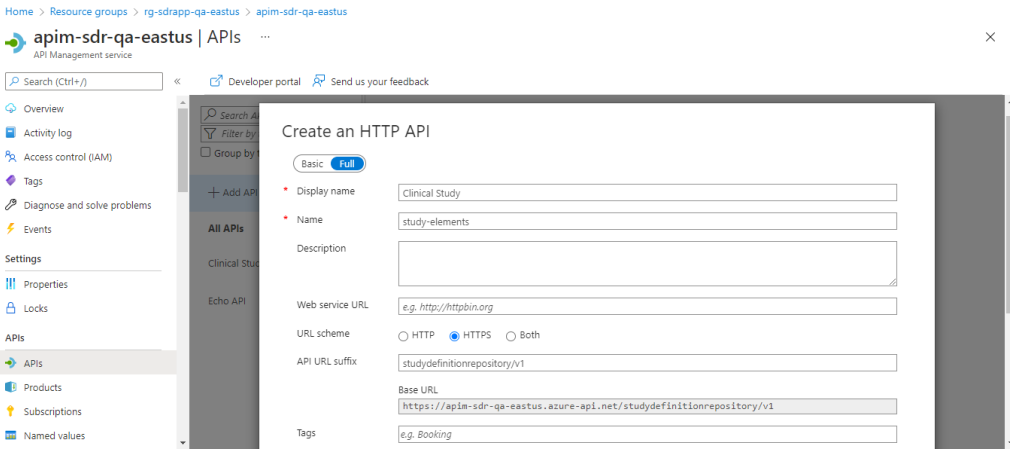
v. In the left navigation of your API Management instance, select APIs > select HTTP API tile.

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-Audiance	Provide the Azure App Registration Scope URL, Refer to Section 2.8.2 step 6 for scope URL (E.g.: api://0000-0000-000-000/ui-access)
AzureAd-Authority	Provide the Azure Ad authority value (https://login.microsoftonline.com/(Provide Azure AD Tenant ID))

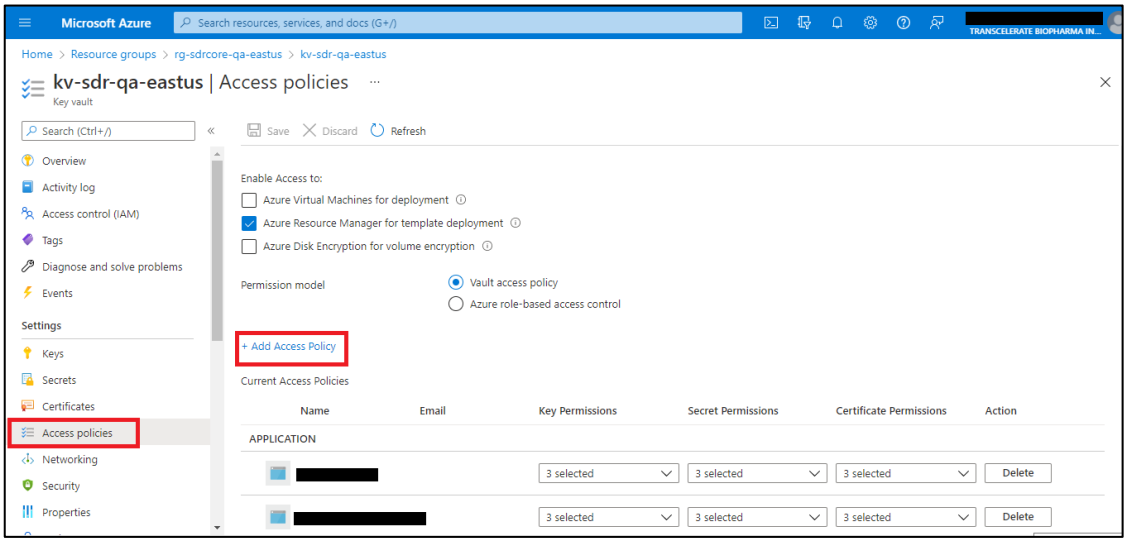
AzureAd-ClientId	Provide the App Registration client ID, refer to Section 2.8.2 App Registration step 11 for client ID
AzureAd-LoginUrl	Provide the Front End (UI) App Service URL.
AzureAd-RedirectUrl	Provide the Redirect URL (E.g.: https://Front End App service URL (UI)/home)
AzureAd-TenantId	Provide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

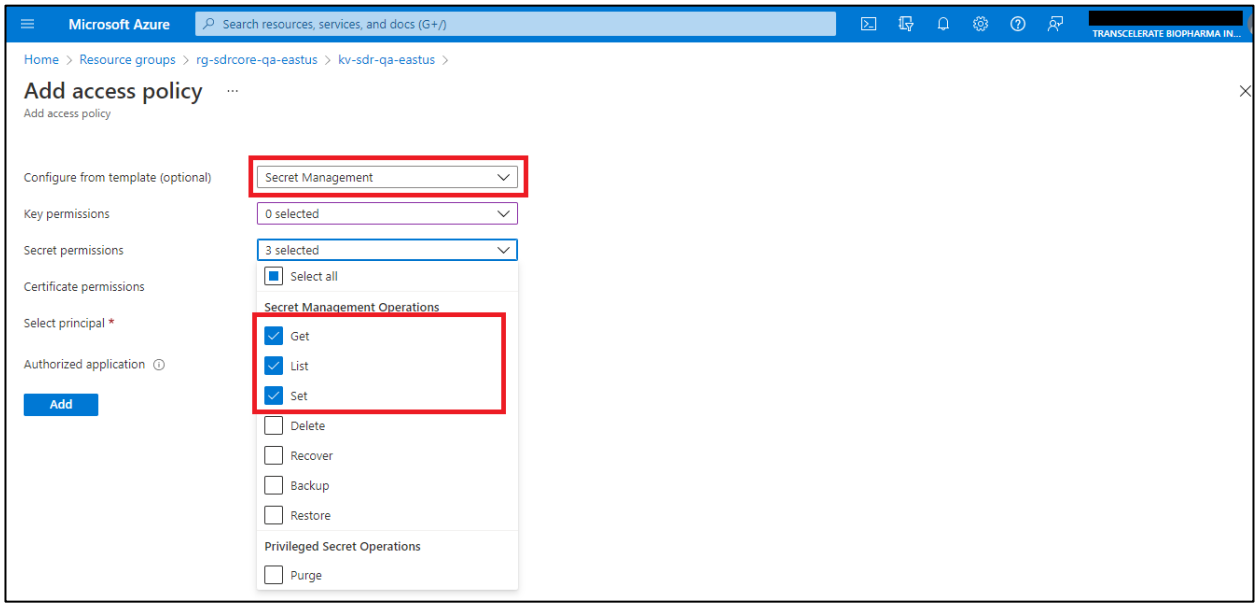
i. Go to → Key Vault → Select Access Policies → Click on Add Access Policy

Figure 33 Add Key Vault Access Policy



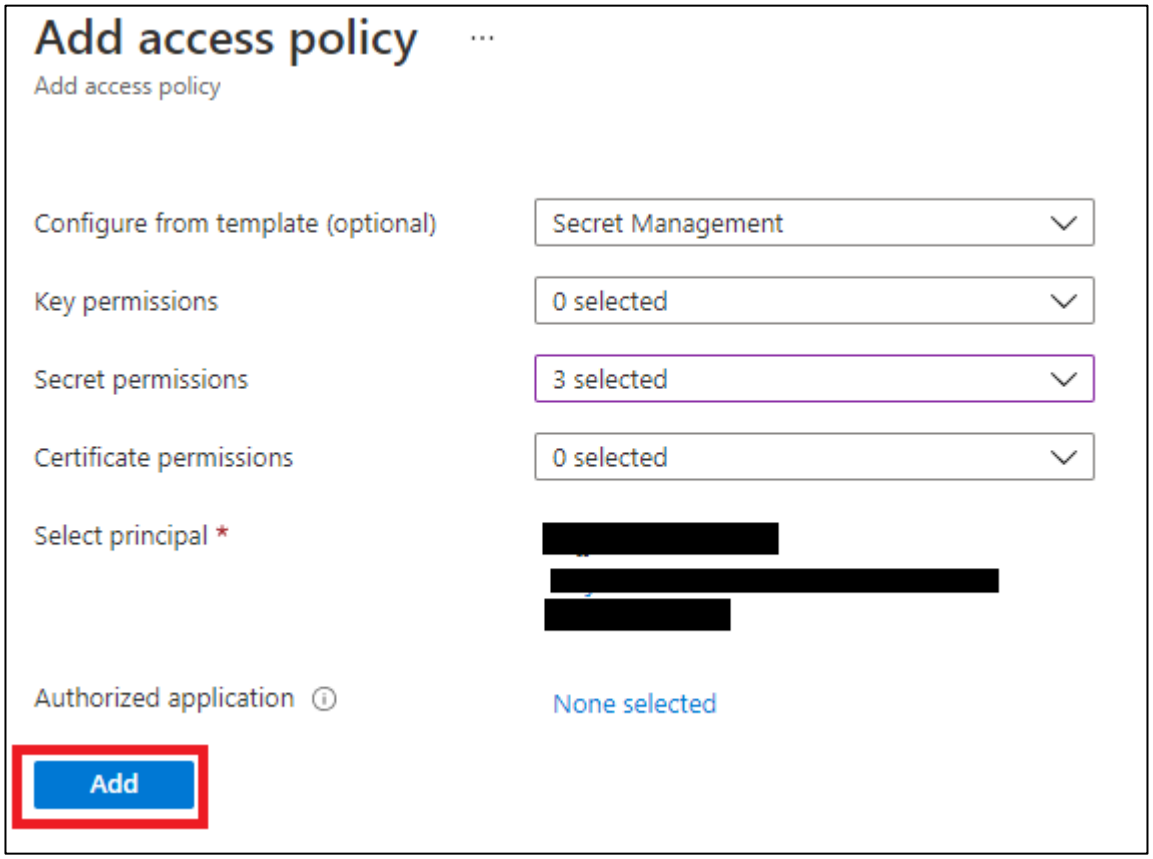
ii. Select Secret Management → select Secret Permissions (Get, List, Set)

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

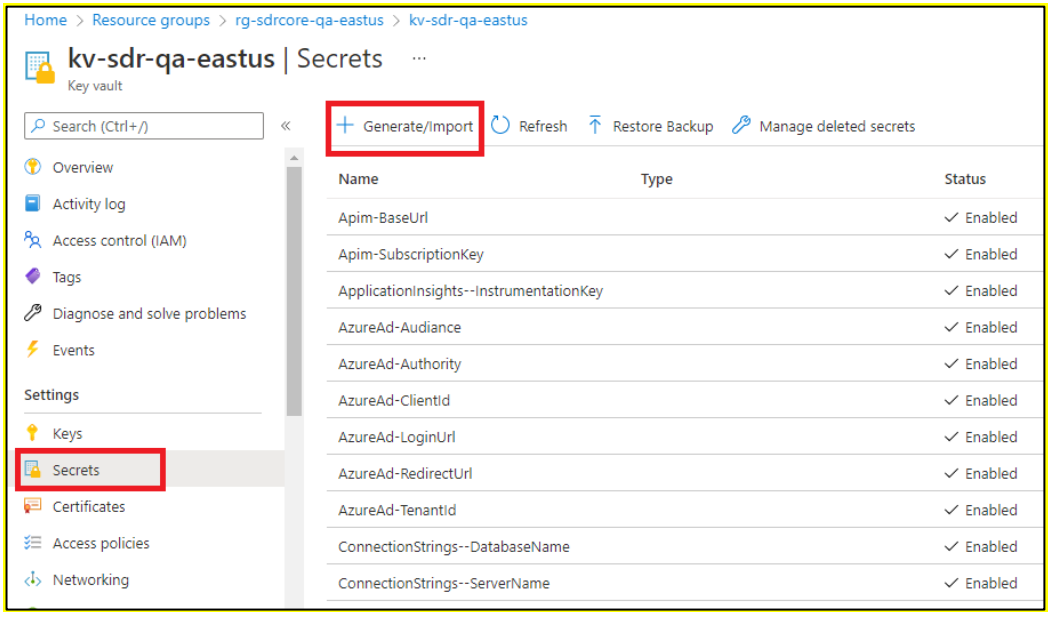


STEPS FOR ADDING KEY VAULT SECRETS:

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

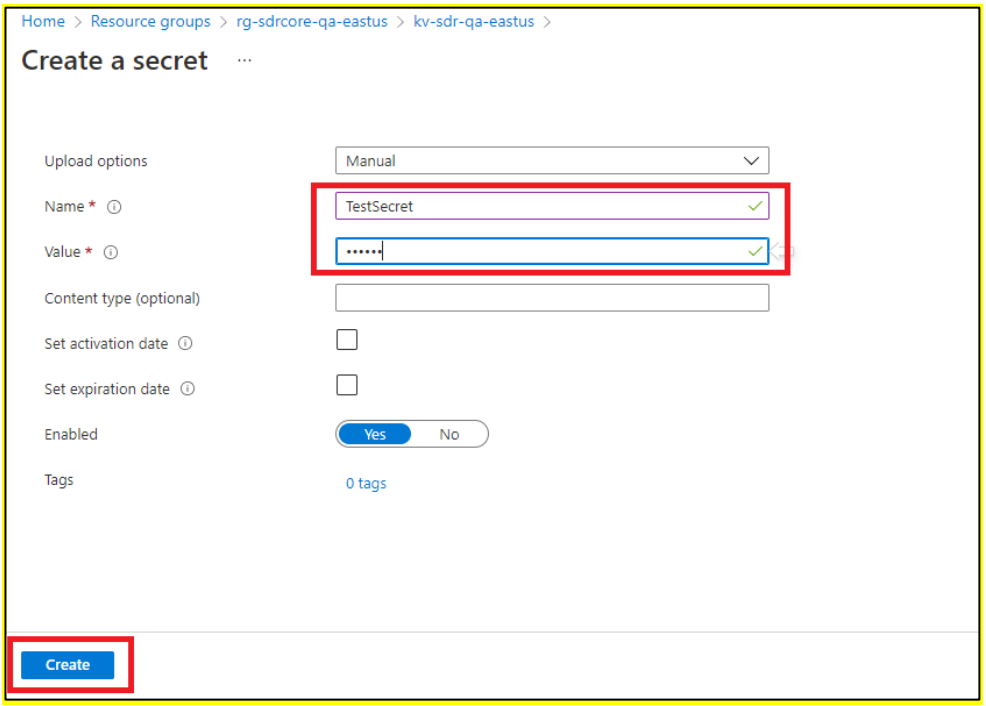
i. Go to → Key Vault → Select Secrets → Click Generate/Import

Figure 36 Create Secrets in Key Vault



ii. Provide Secret Name and Value → Select Create

Figure 37 Add Secrets values in Key Vault



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.

Low Level Design Document

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: e*astus, westus, etc.*

Figure 38 Resource Naming Convention

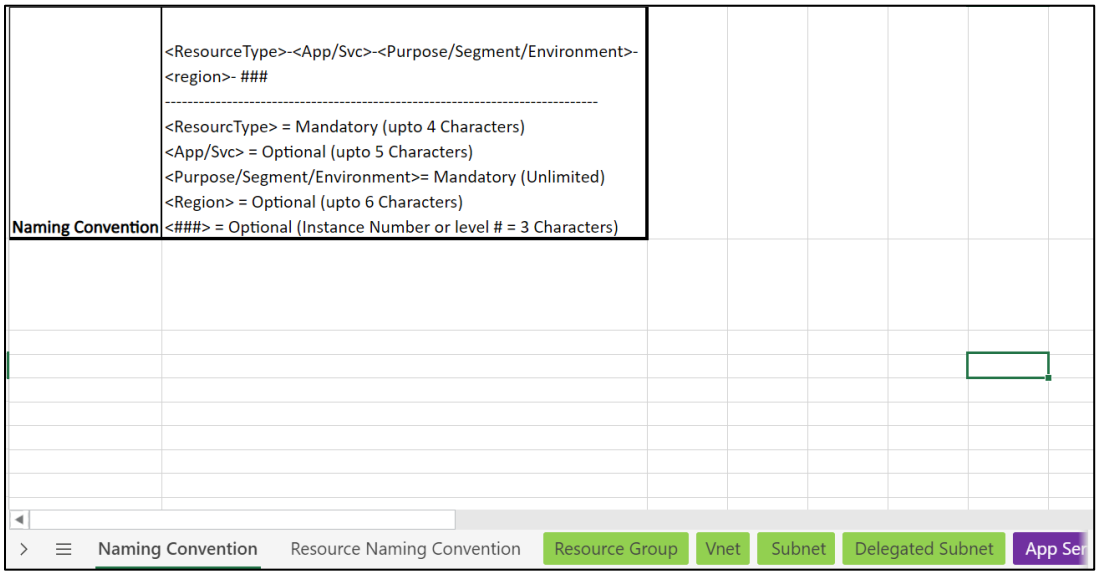
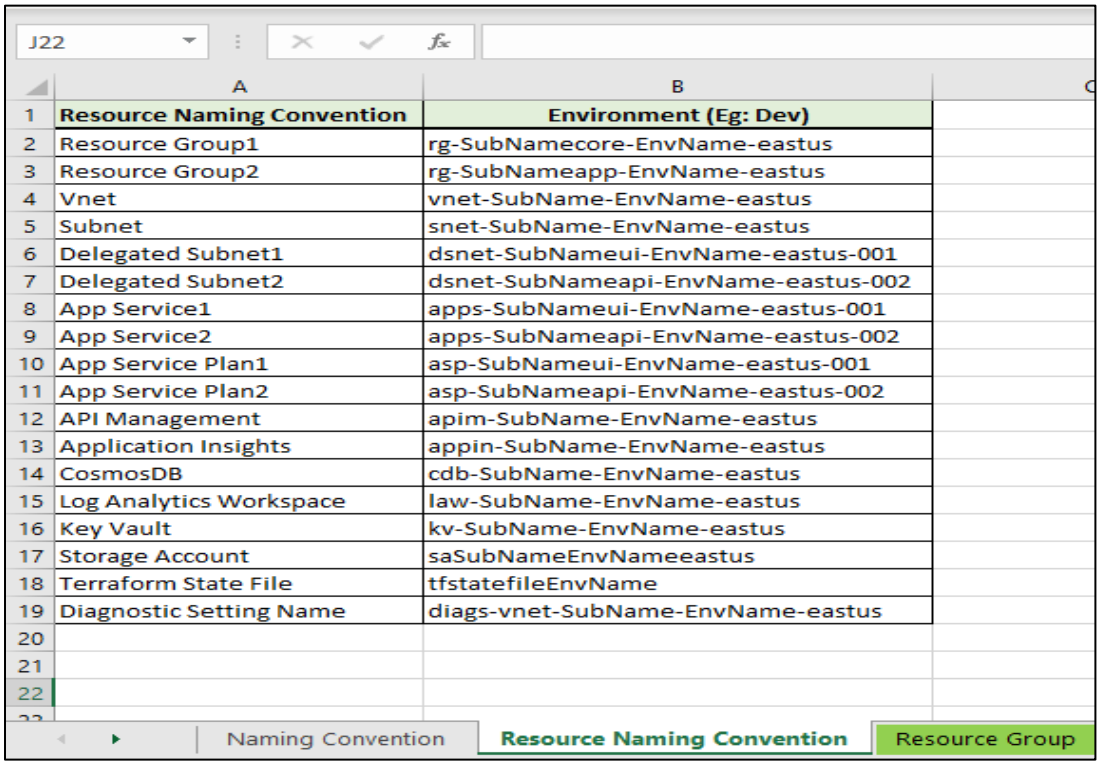


Figure 39 SDR Resource Naming Convention



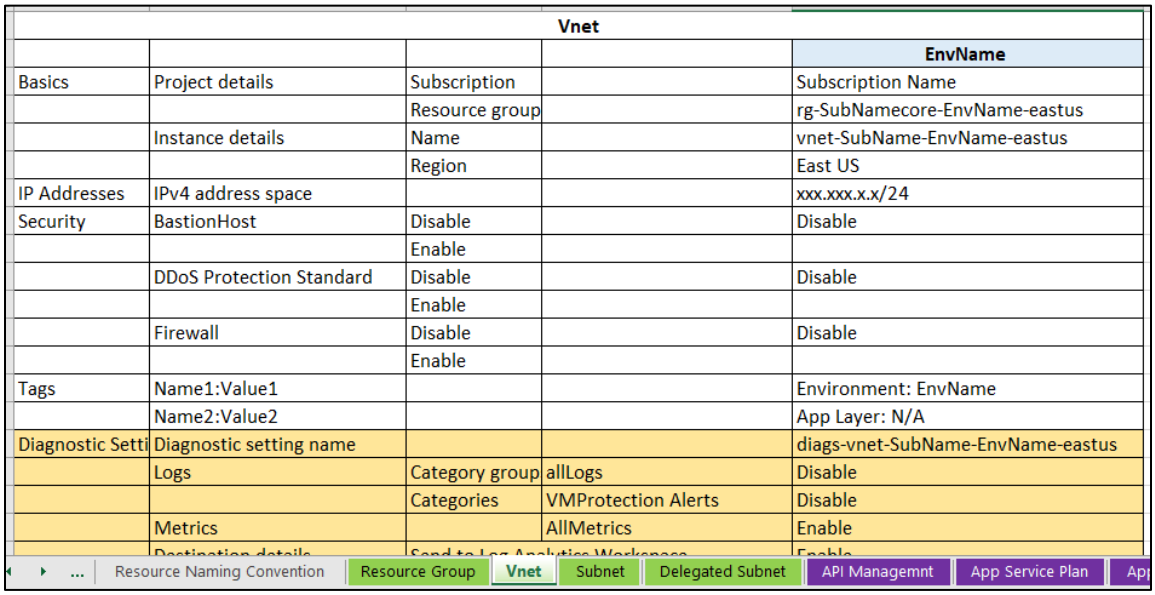
	A	B
1	**Resource Naming Convention**	**Environment (Eg: Dev)**
2	Resource Group1	rg-SubNamecore-EnvName-eastus
3	Resource Group2	rg-SubNameapp-EnvName-eastus
4	Vnet	vnet-SubName-EnvName-eastus
5	Subnet	snet-SubName-EnvName-eastus
6	Delegated Subnet1	dsnet-SubNameui-EnvName-eastus-001
7	Delegated Subnet2	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 Setti	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 Managemnt | App Service Plan | Ap

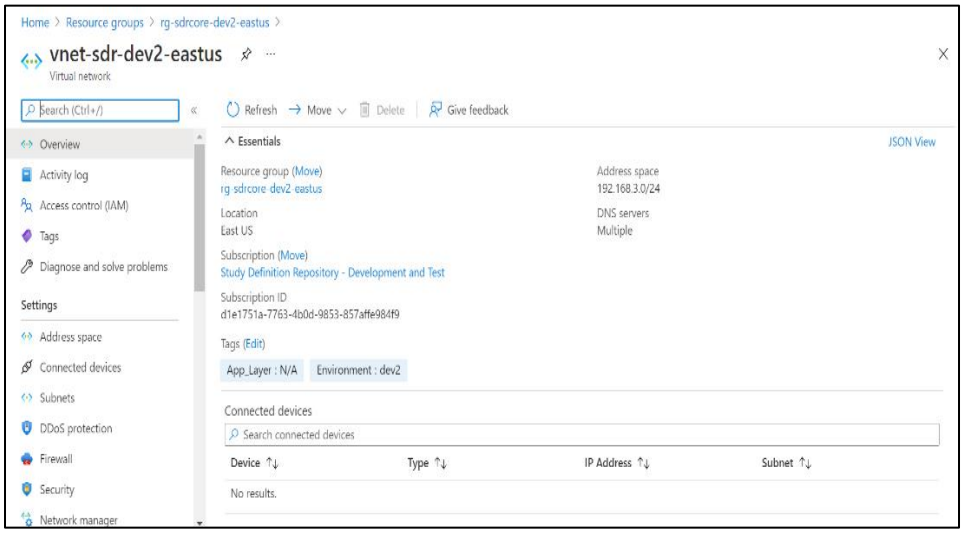
PRE-REQUISITES:

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

STEPS

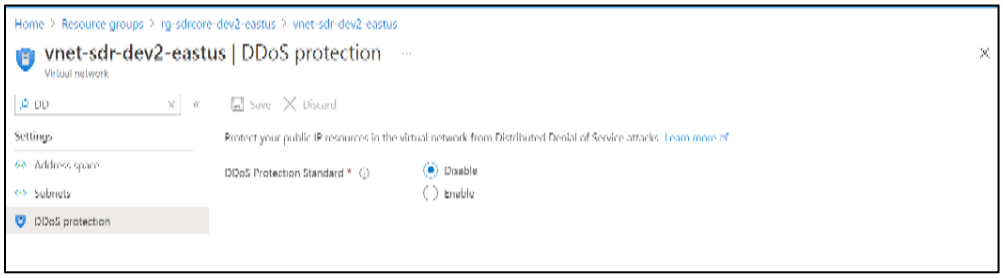
i. Login to Azure Portal
ii. Click on the Resource Groups tab
iii. Select the Resource group for VNet configuration
iv. Verify that the Basic details like Subscription, Resource Group, Name and Region is as per the LLD
v. Verify that the IPv4 Address Space is as per the LLD

Figure 41 Virtual Network



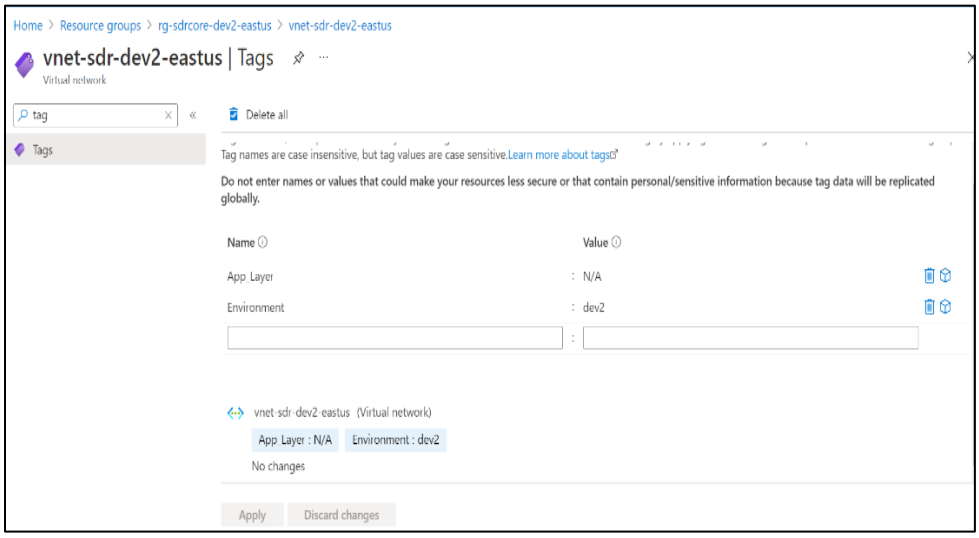
vi. Go to Security tab and verify that Bastion Host is set as per the LLD
vii. Go to DDoS Protection tab and verify that it is set as per the LLD

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

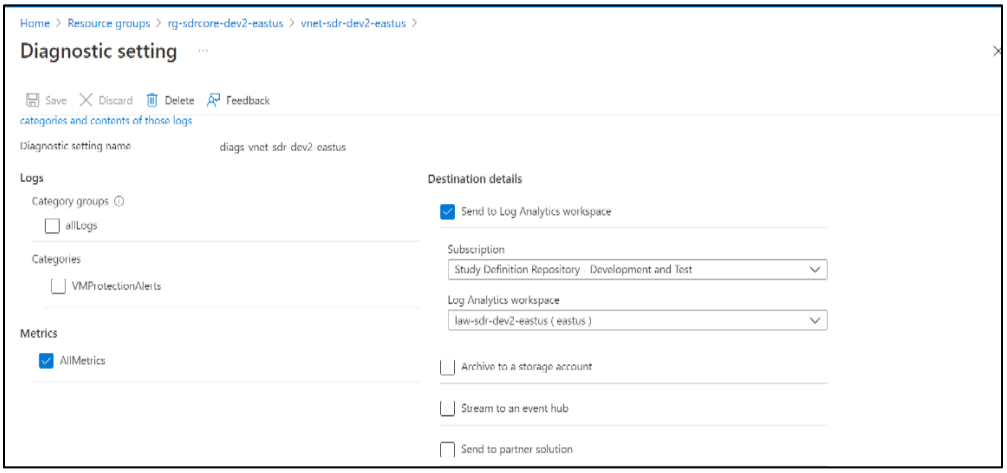
Figure 43 Virtual Network - Tags



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

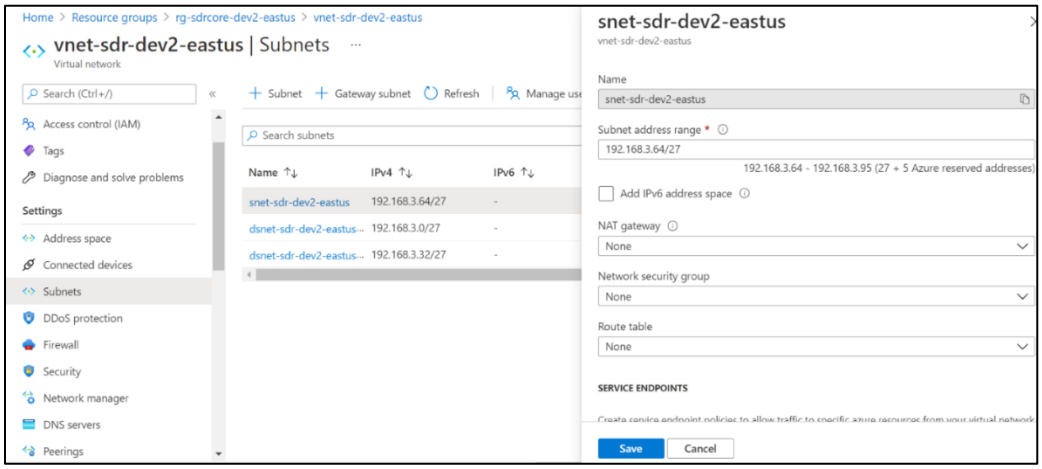
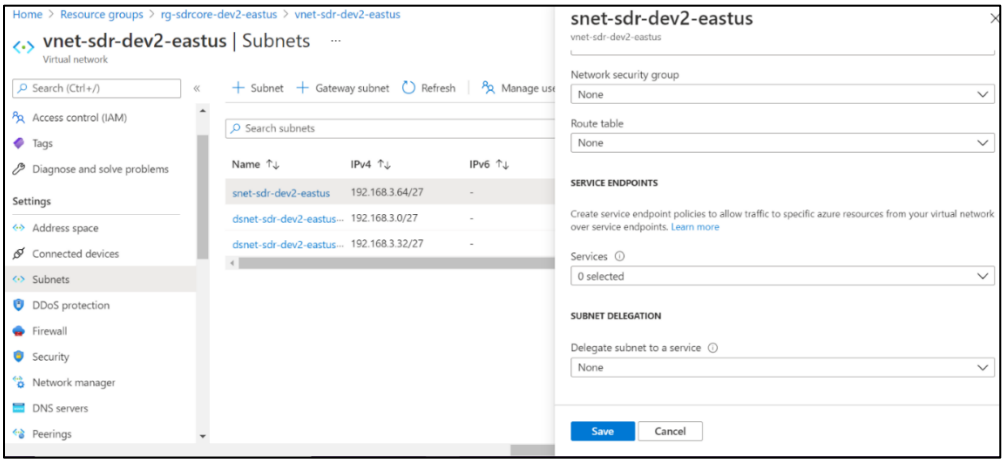


Figure 46 Subnet Details



3.4. Delegated Subnet

Validation of Delegated Subnet configuration.

PRE-REQUISITES:

- Reader level of access at Resource group Level.

STEPS

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

Figure 47 Delegated Subnet-001 Details

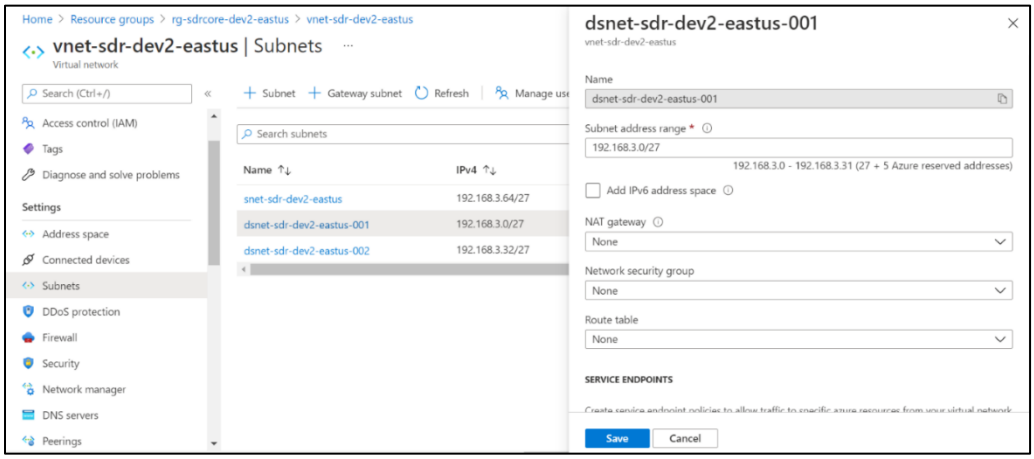


Figure 48 Delegated Subnet-001 Service Endpoints Details

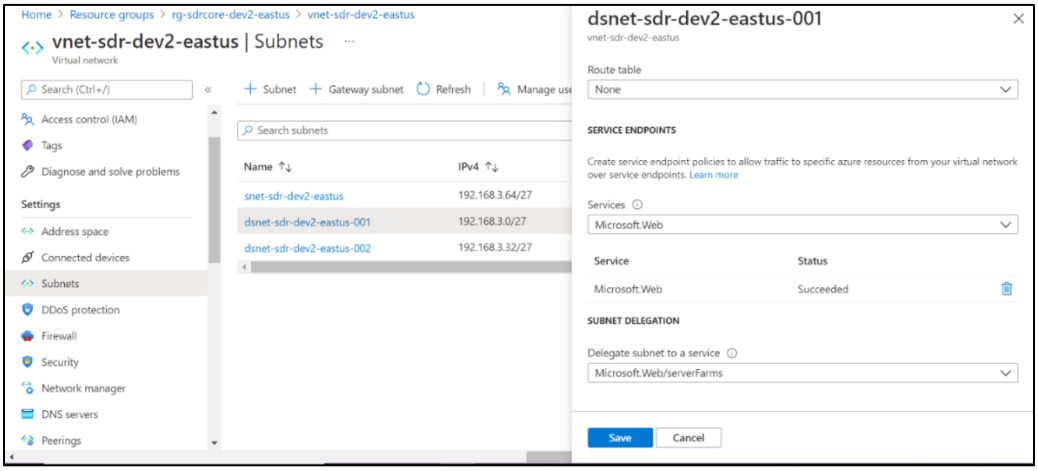


Figure 49 Delegated Subnet-002 Details

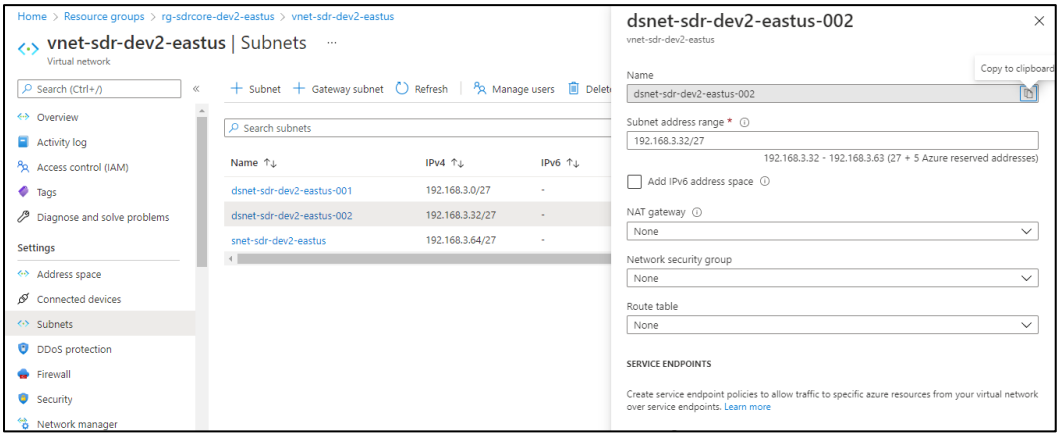
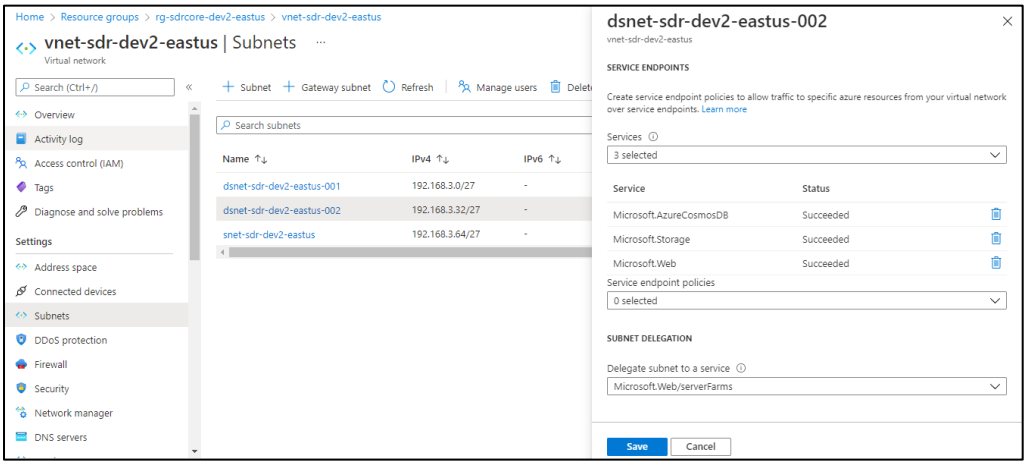


Figure 50 Delegated Subnet-002 Service Endpoints Details



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

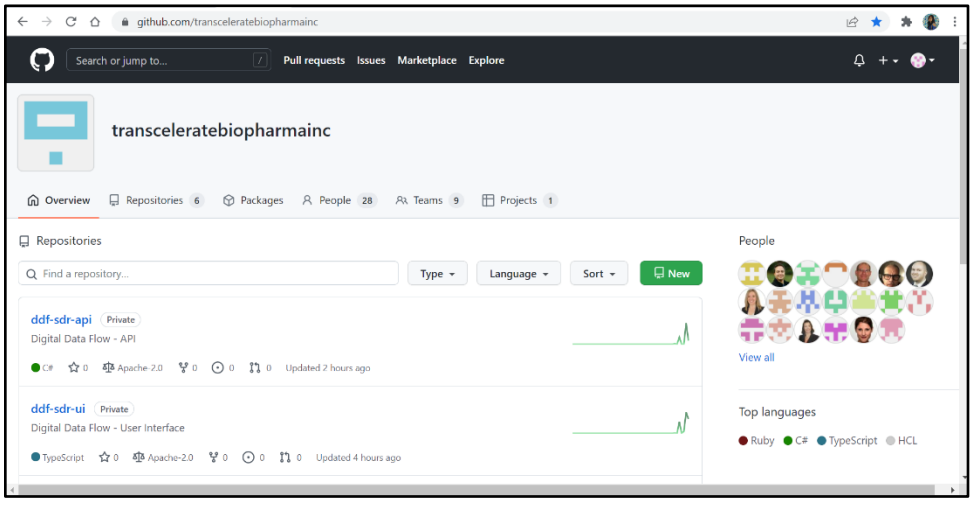
PRE-REQUISITES

- Contributor level of access at Resource Group level.

DEPLOYMENT STEPS:

i. Go to https://github.com/transceleratebiopharmainc, GitHub URL.

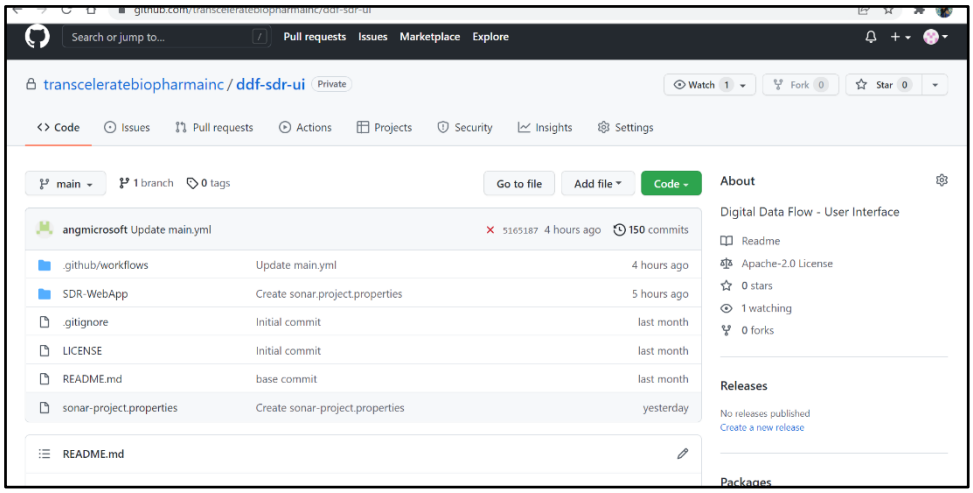
Figure 51 TransCelerate GitHub Project



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

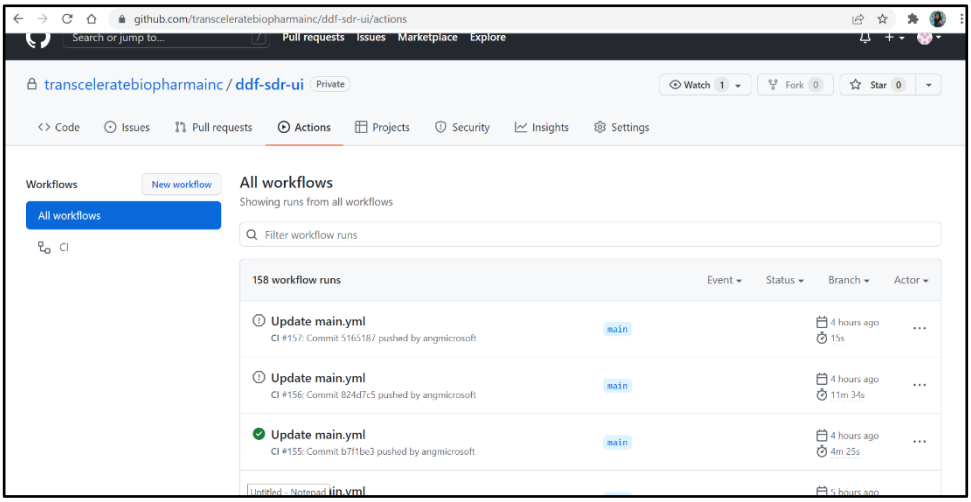
https://github.com/transceleratebiopharmainc/ddf-sdr-ui

Figure 52 TransCelerate GitHub SDR UI Repo



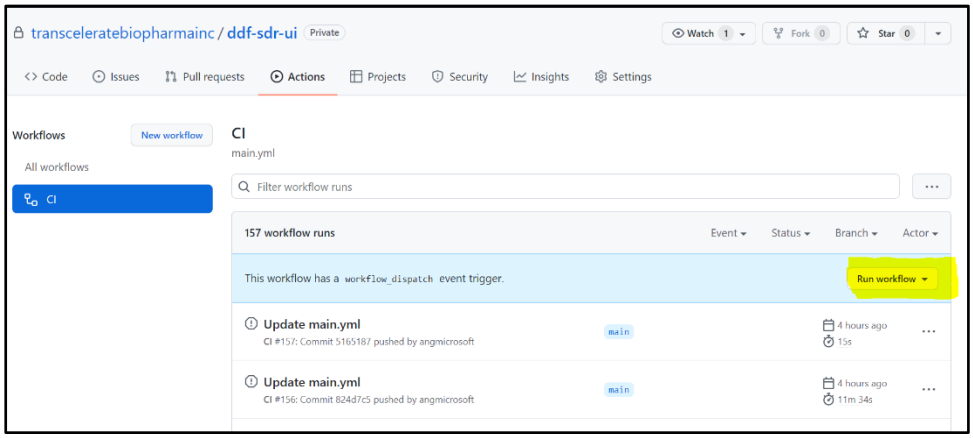
iii. Click on Actions tab.

Figure 53 SDR UI Repo - GitHub Actions



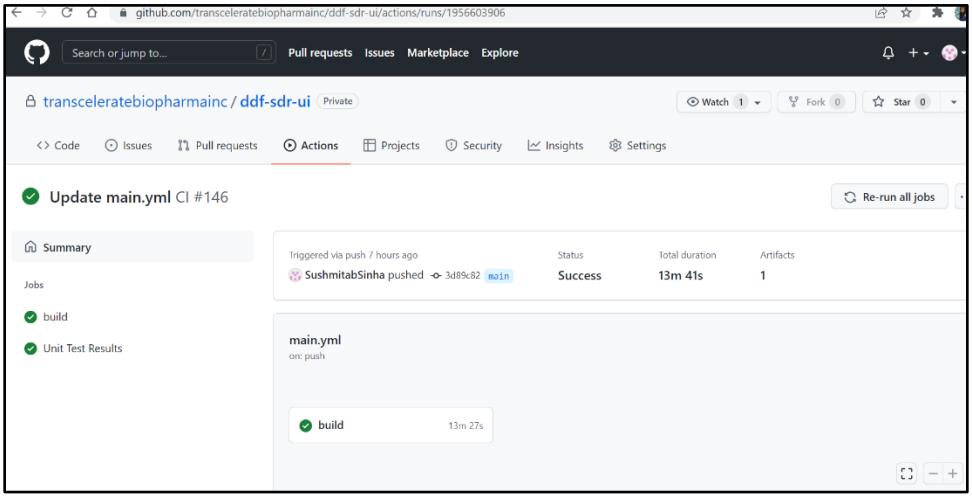
iv. Click on the workflow CI under All workflow.

Figure 54 GitHub Actions - CI Workflow



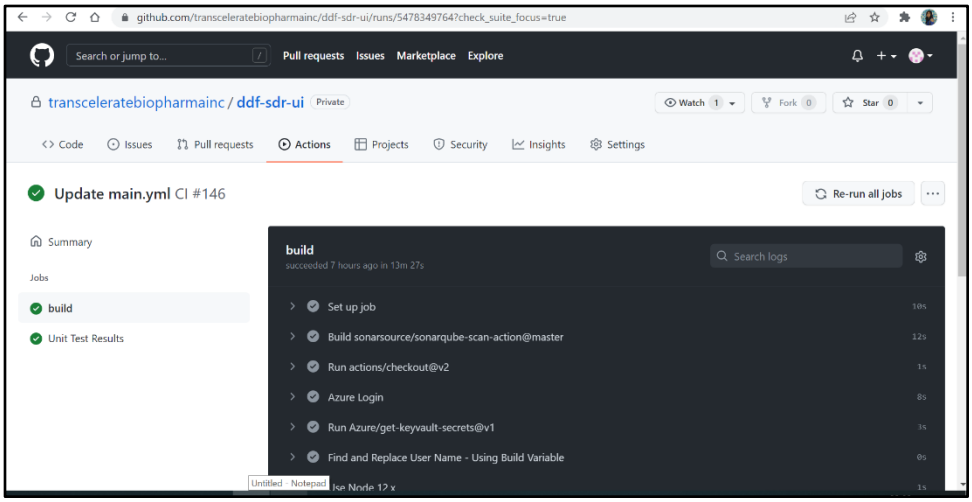
v. Click on Run Workflow on the right-hand side. Then the action will be triggered.

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

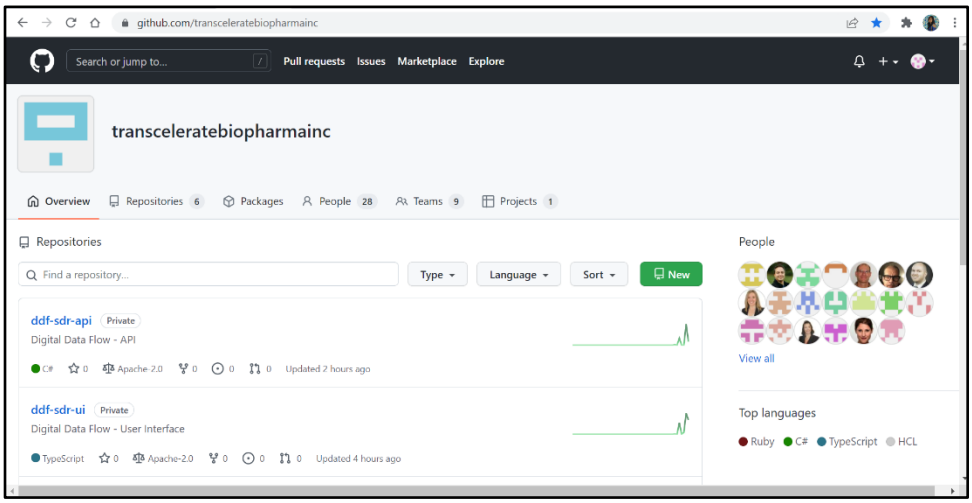
PRE-REQUISITES

- Contributor access at Resource group Level.

DEPLOYMENT STEPS:

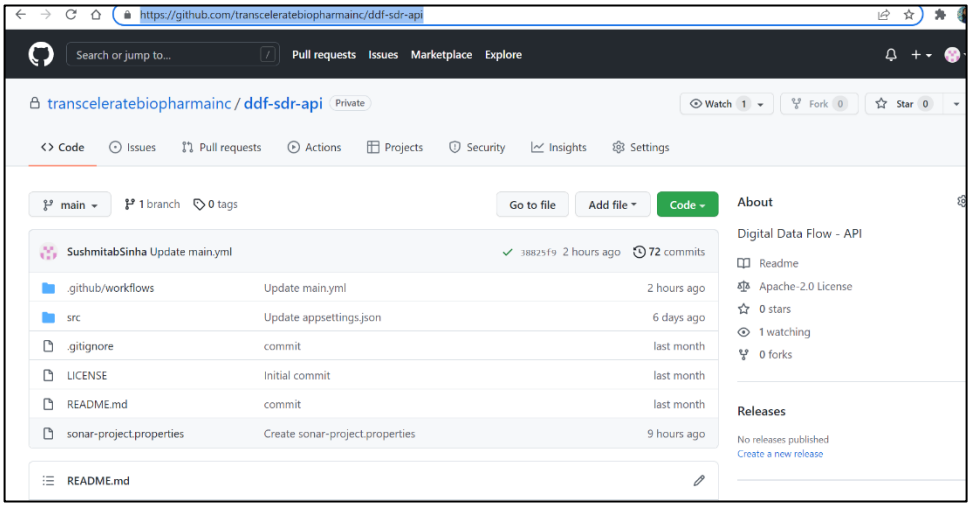
i. Go to https://github.com/transcelerateBiopharmainc, GitHub URL.

Figure 57 TransCelerate GitHub Project



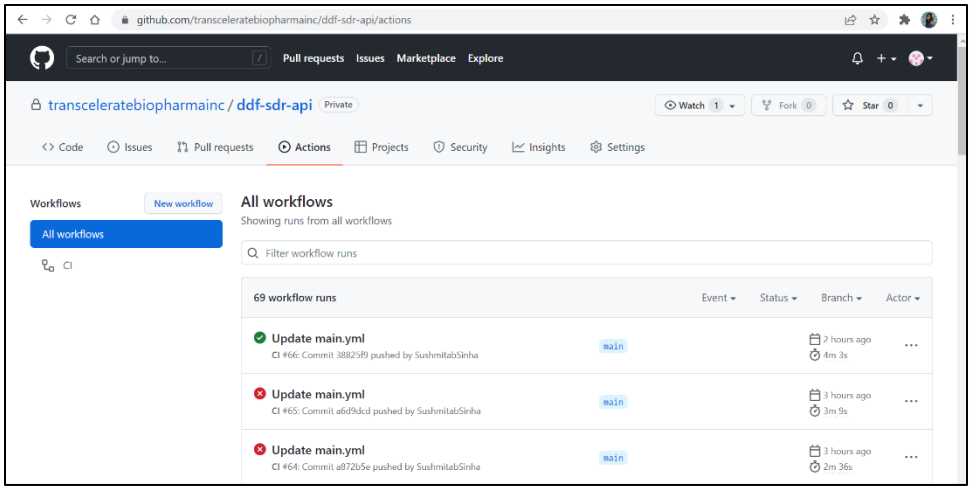
ii. Select the required repository, here **ddf-sdr-api.**
URL - https://github.com/transceleratebiopharmainc/ddf-sdr-api

Figure 58 TransCelerate GitHub SDR API Repo



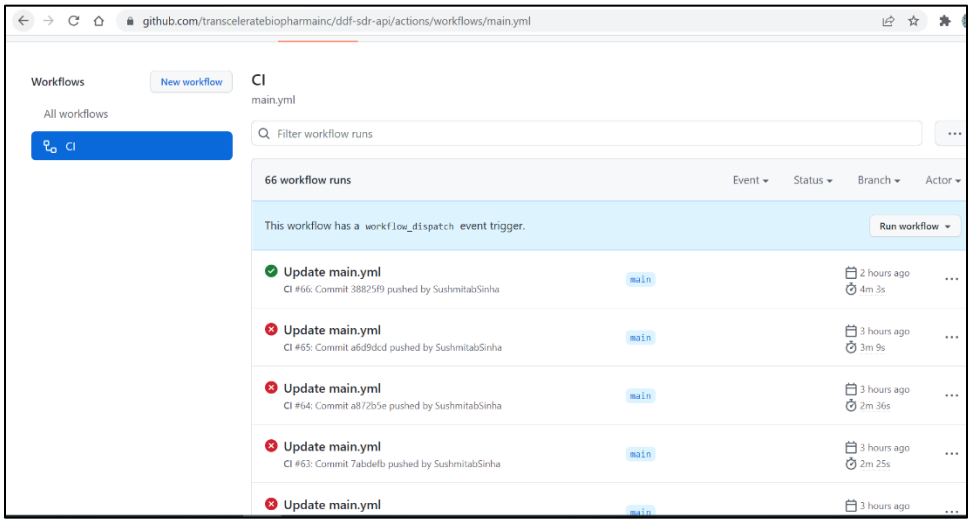
iii. Click on Actions tab.

Figure 59 GitHub Actions CI Worklfow



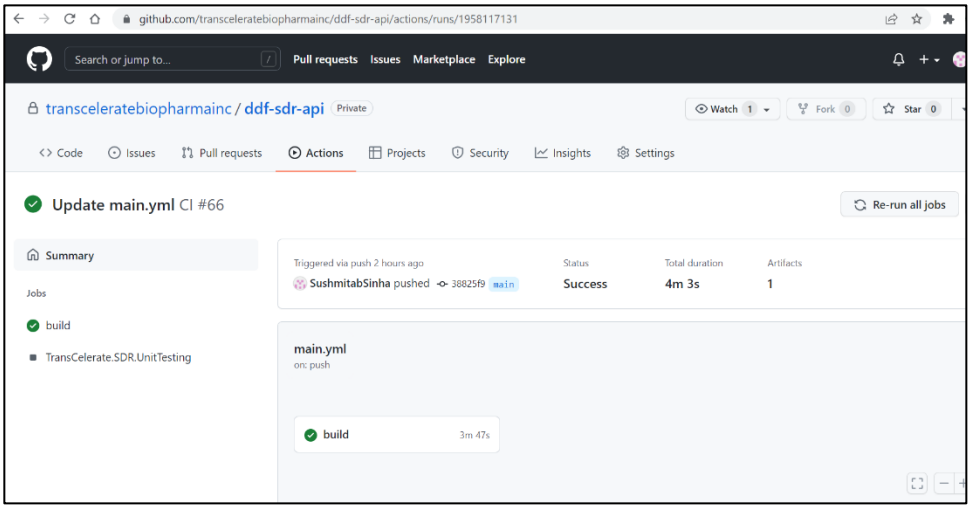
iv. Click on the workflow CI under All workflow.

Figure 60 GitHub CI Workflow Run



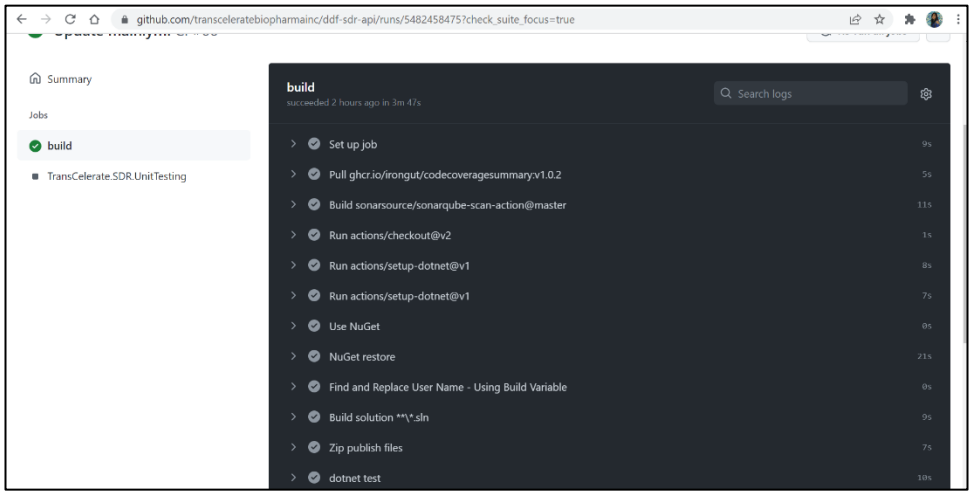
v. Click on Run Workflow on the right-hand side. Then the action will be triggered.

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.

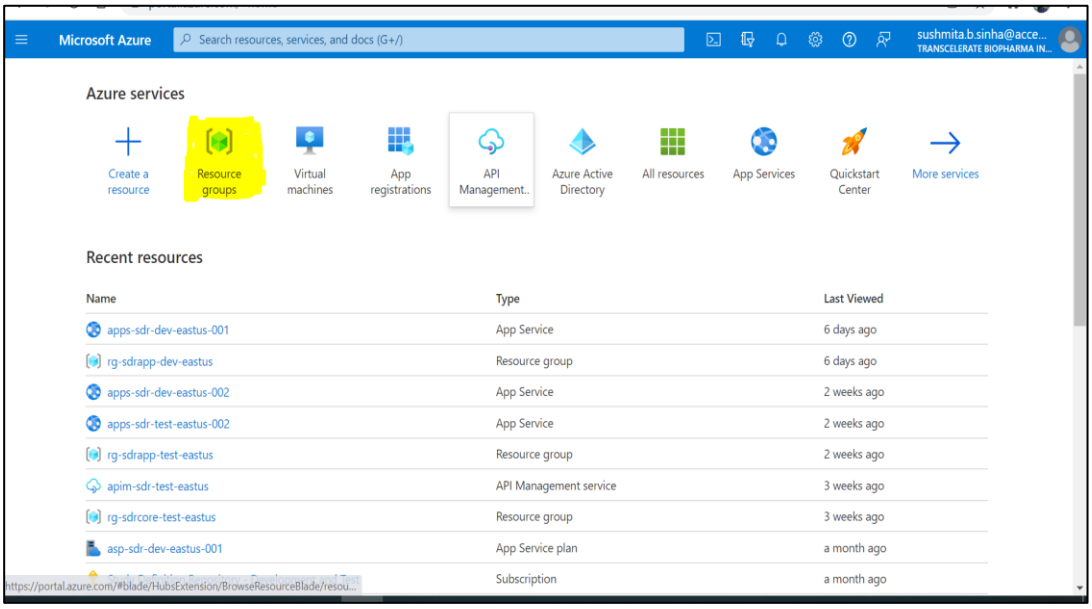
4.4 Deployment Verification

UI APPLICATION VERIFICATION STEPS:

This is to verify the UI Application deployment was successful.

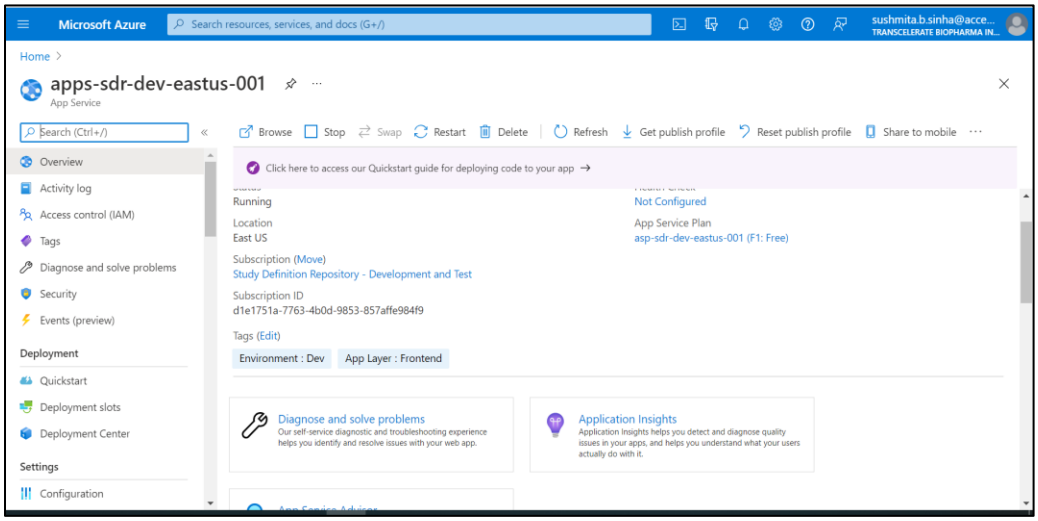
i. Go to portal.azure.com. Click on resource group.

Figure 63 Azure Portal



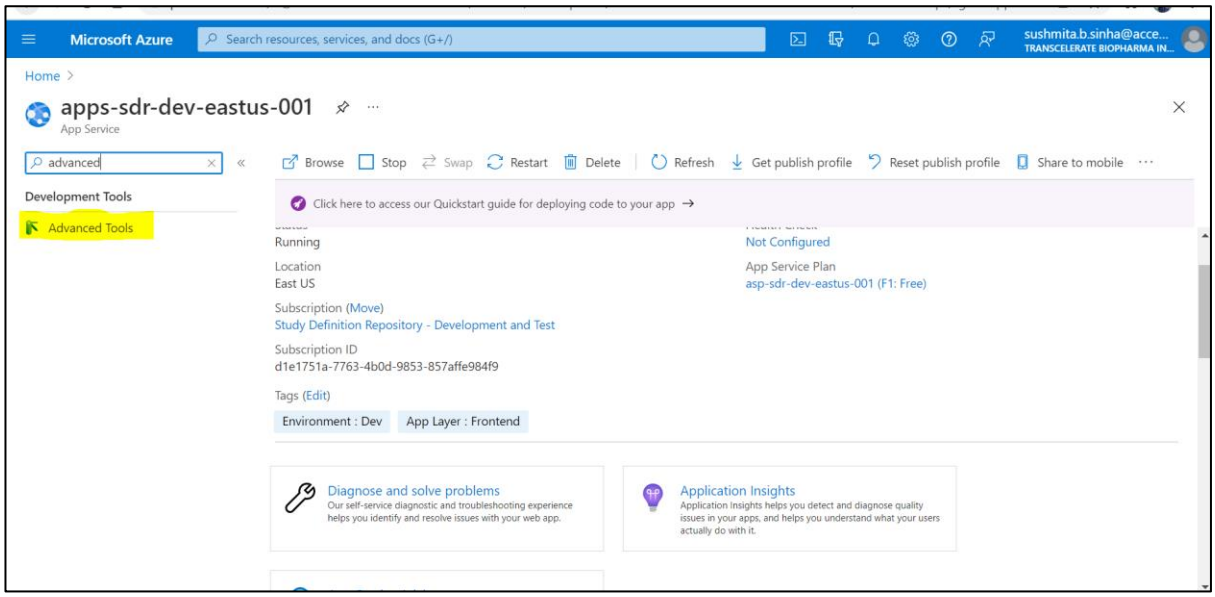
ii. Select the required resource group and select the UI App Service instance.

Figure 64 Azure Portal - SDR UI App Service



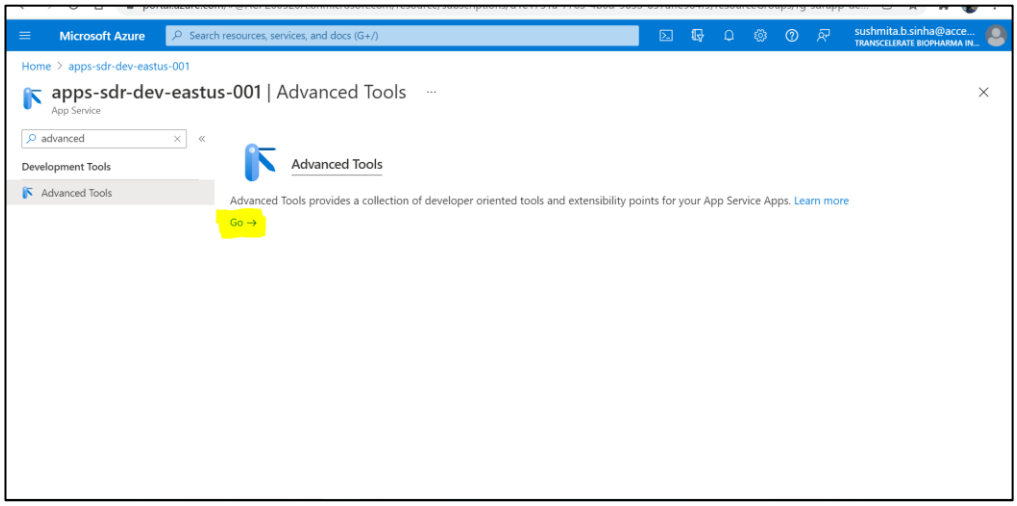
iii. In search box, search for Advanced tools.

Figure 65 SDR UI App Service - Advanced Tools



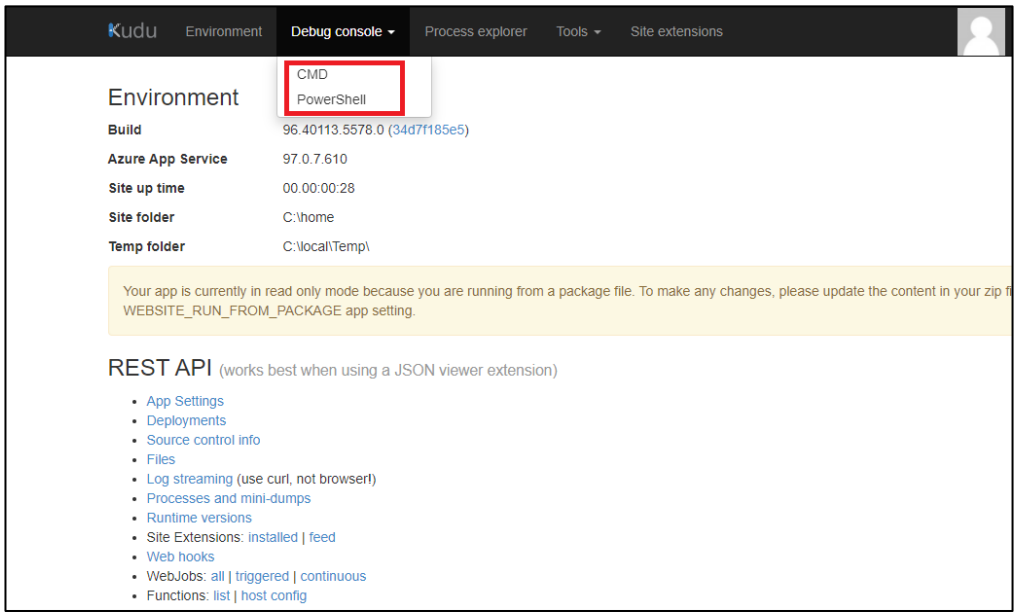
iv. Click on Go.

Figure 66 SDR UI App Service Advanced Tools - Go



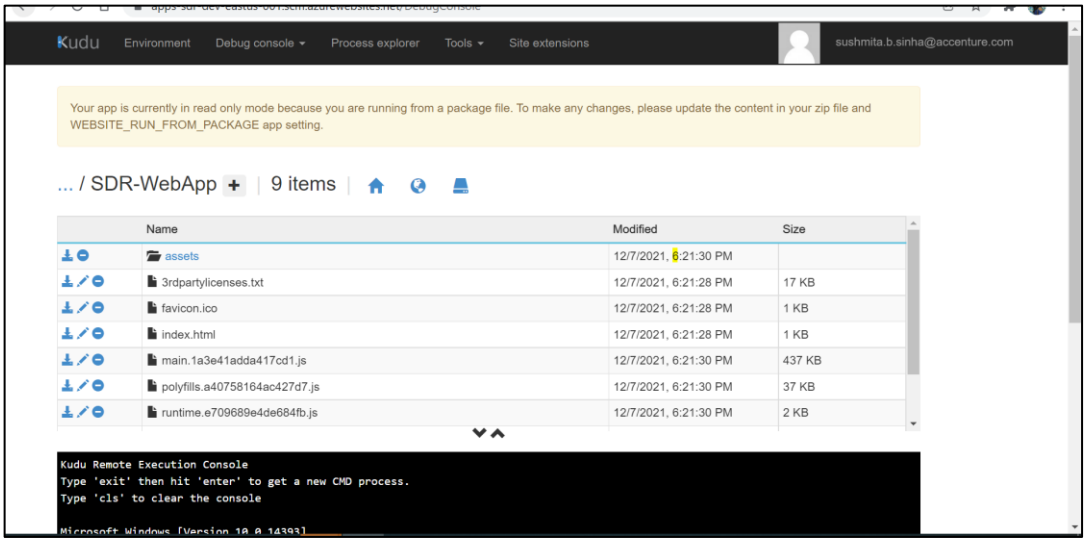
v. Go to Debug console and select CMD/Power Shell

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)

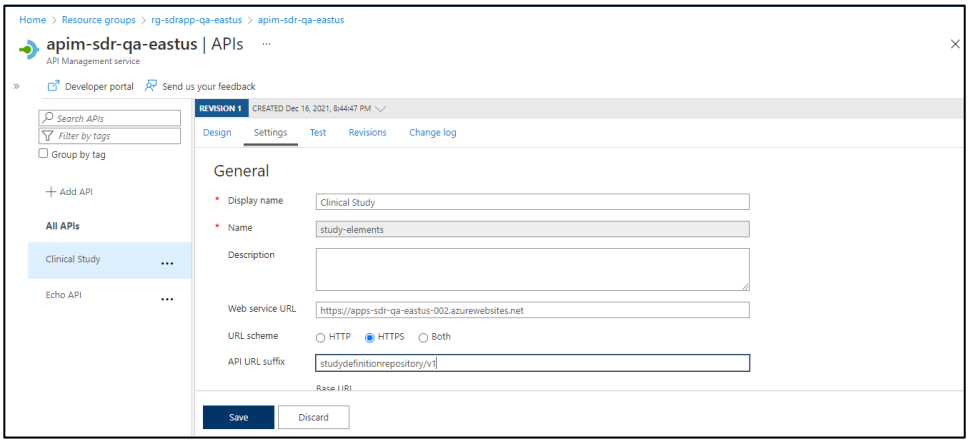
PRE-REQUISITES:

- Contributor access at Resource Group level.

STEPS:

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

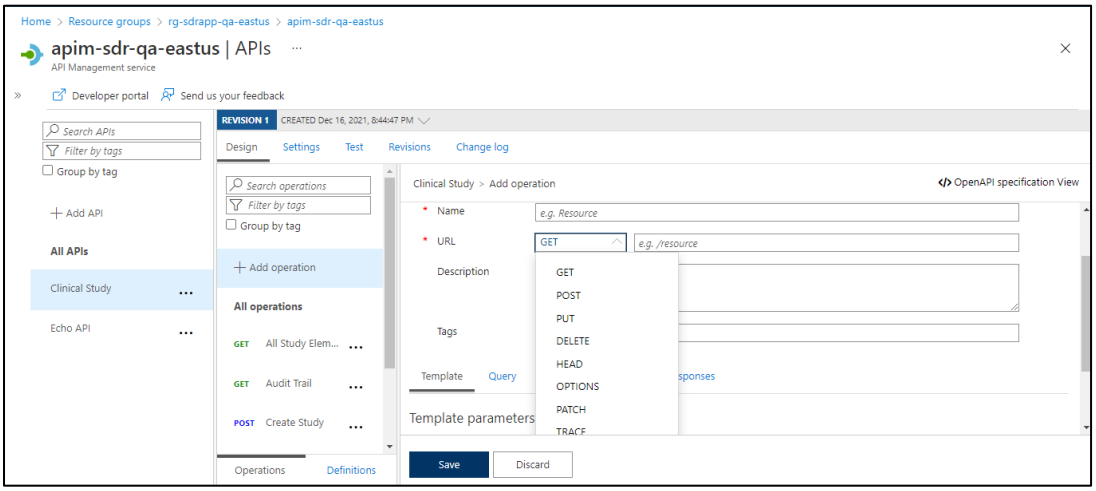
Figure 69 API Management - Name HTTP API



- 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

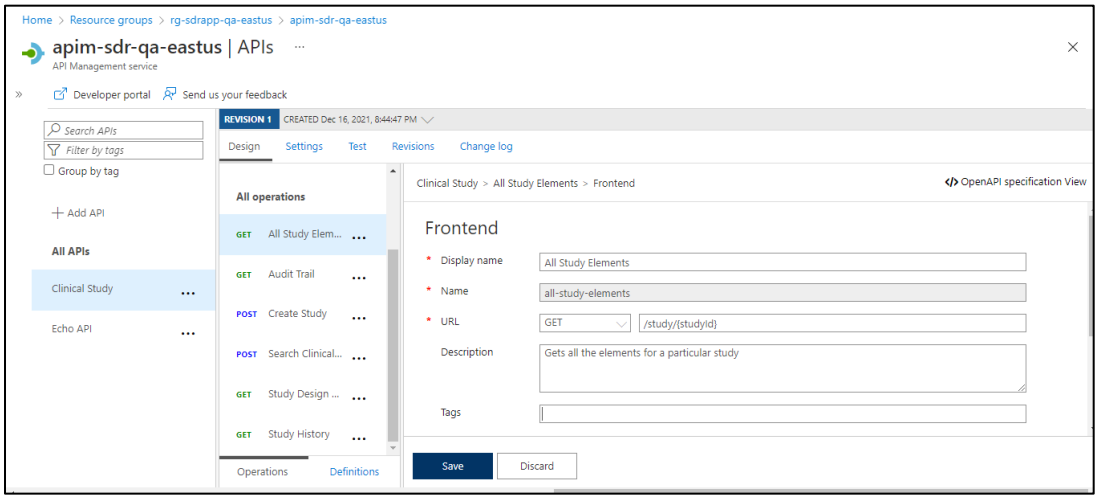
ii. Go to design tab and select an operation for the API.

Figure 70 API Management - HTTP API Operation



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

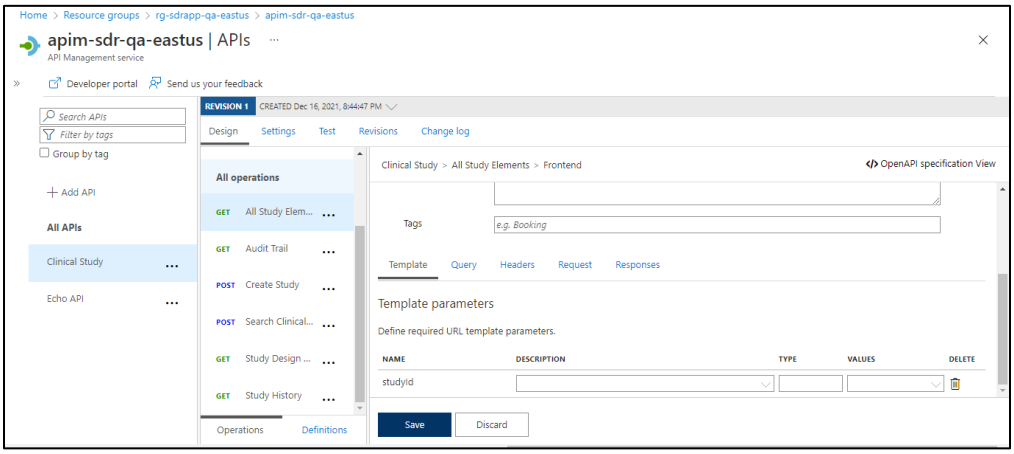
Figure 71 API Management - API Operation Details



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

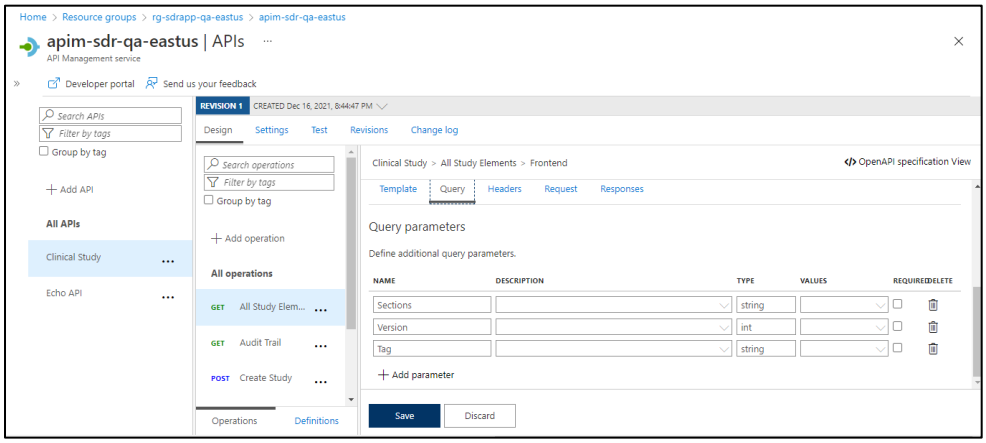
iv. Click on template > add parameter – study

Figure 72 API Management – API Operation - Add Parameter



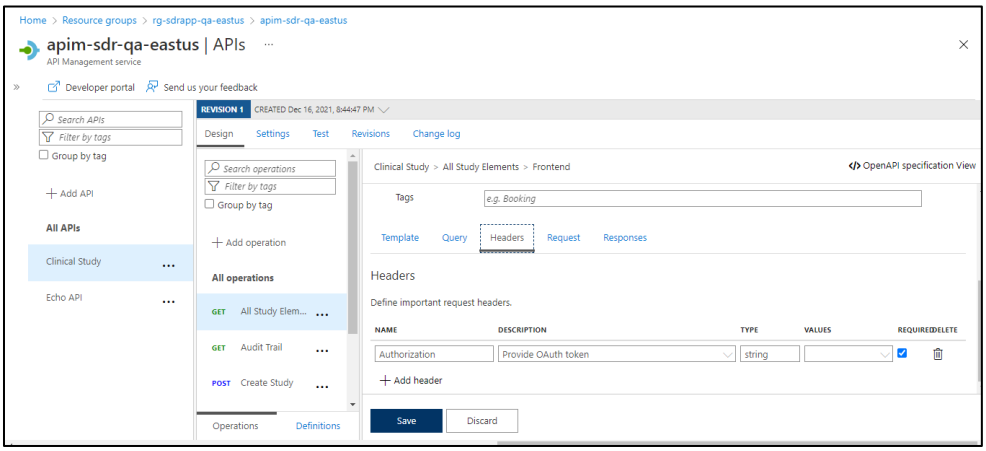
v. Click on Query > Add Query parameters as below.

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	Sections	string	Authorization	string
			Version	int		
			Tag	string		
Audit Trail	GET /audittrail/{studyId}	studyId	fromDate	string	Authorization	string
			toDate	string		
Study Design Sections	GET /{studyId}/studydesign/{studydesignId}	studyId	Sections	string	Authorization	string
		studydesignId	Version	int		
			Tag	string		
Study History	GET /studyhistory		fromDate	date	Authorization	string
			toDate	date		
Create Study	POST /study				Authorization	string
Search Clinical Study	POST /search				Authorization	string

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

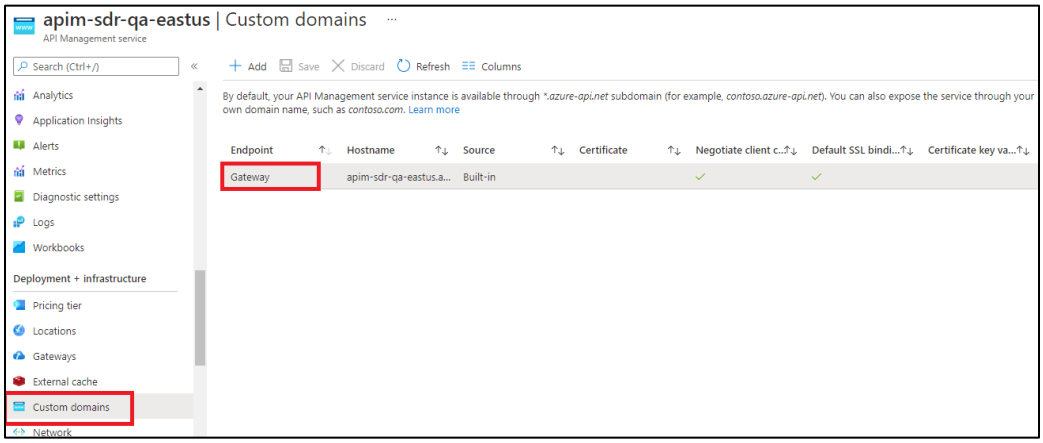
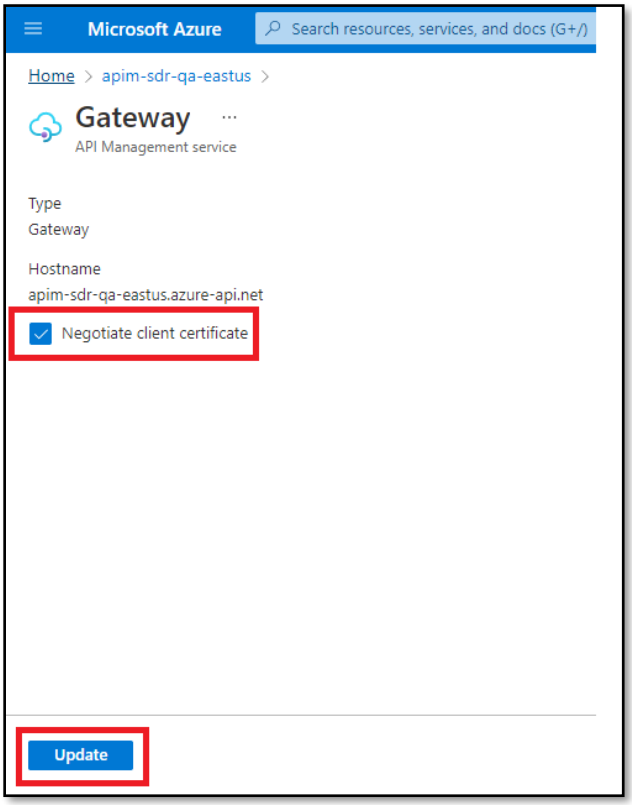


Figure 76 API Management - Custom Domains - Gateway



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
```

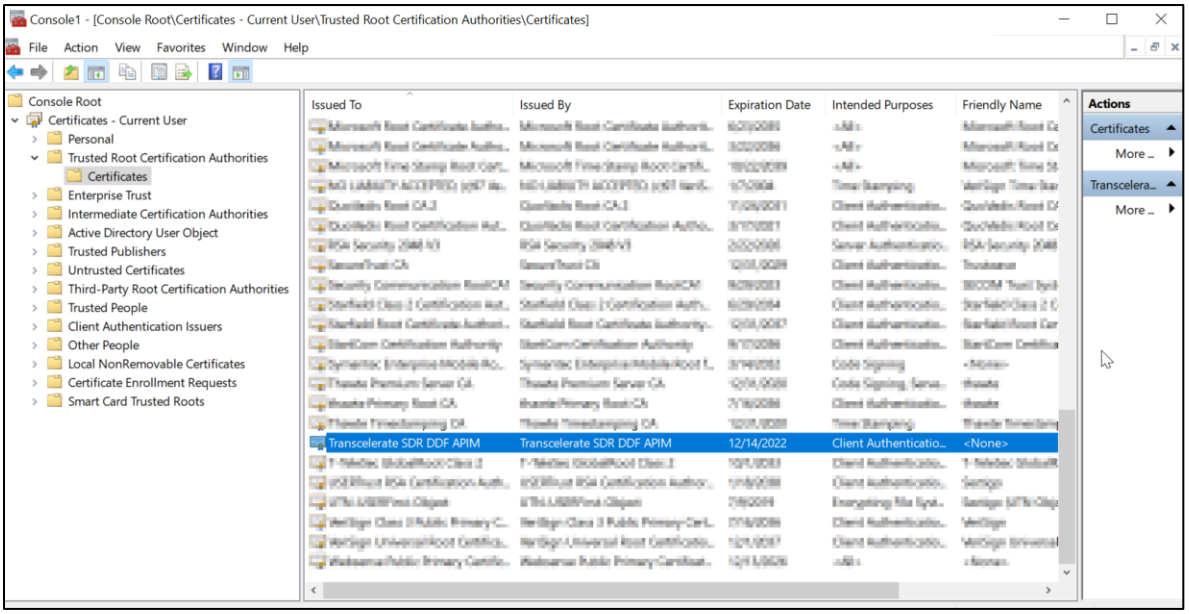
ii. Importing the certificate to trusted publishers on local system

```
#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.

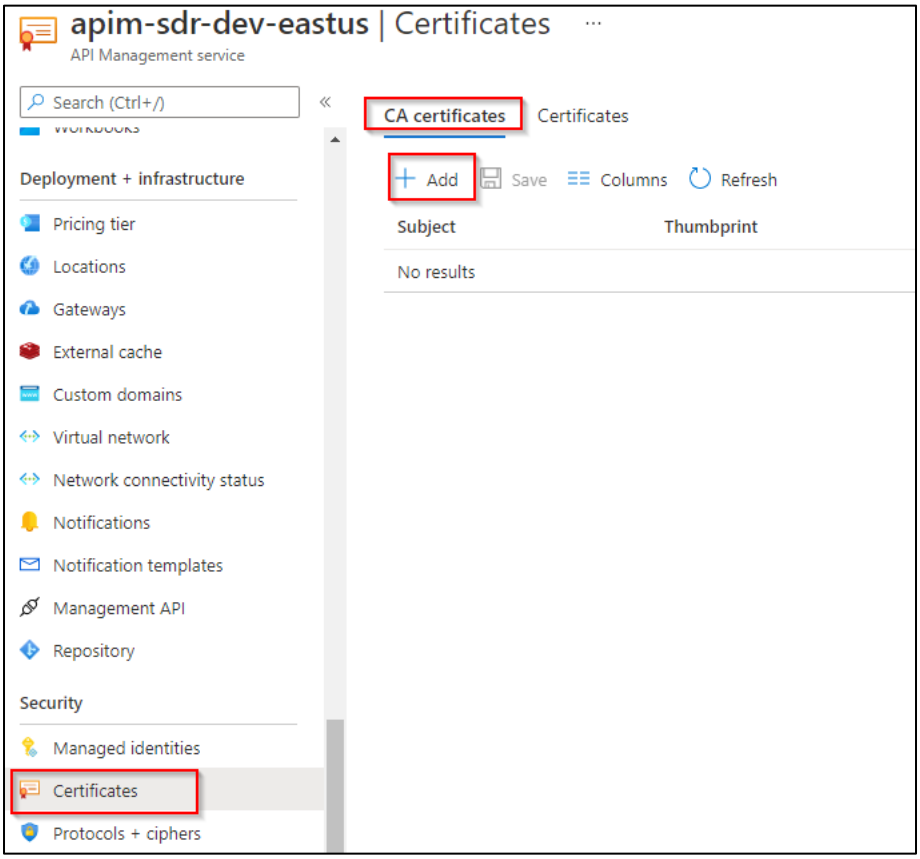
Figure 78 Client Certificate

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

UPLOAD THE CLIENT CERTIFICATE TO APIM:

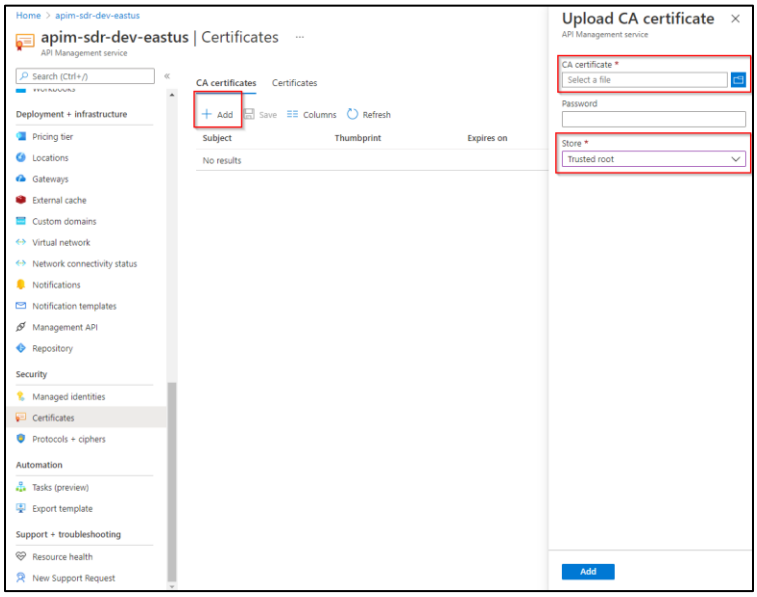
i. In Azure Portal, Go to the Certificates option under the “Security” section of APIM. Go to “CA certificates” option and click on “Add” option

Figure 79 APIM Certificates



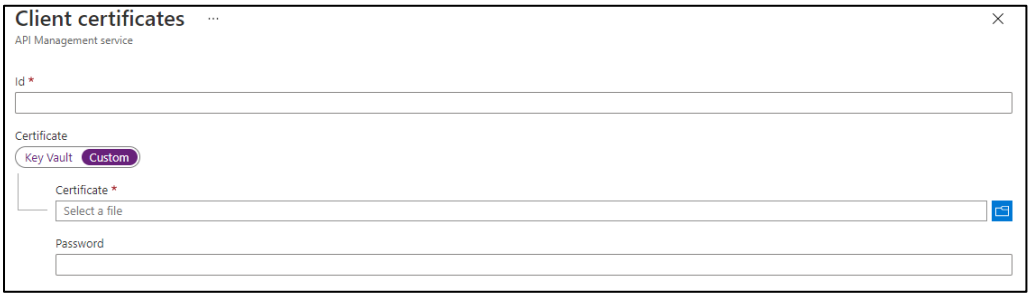
ii. After clicking on the Add option upload the Root CA certificate (.cer) file to the “Trusted root” by selecting the root CA file as below.
Note: Refer

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



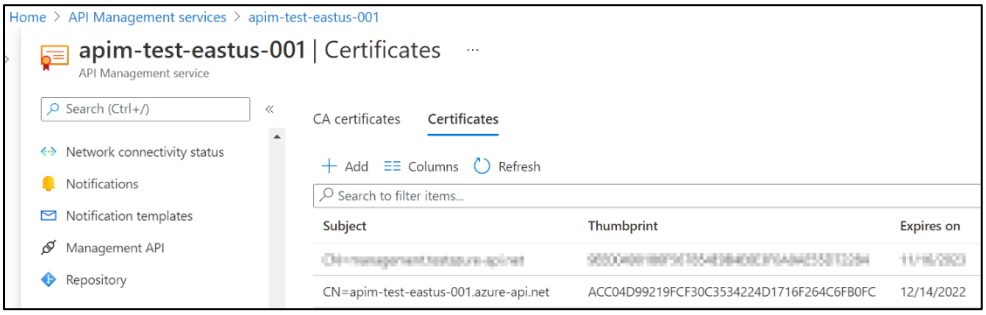
iv. Once both the certificates are uploaded it should be visible on the API Management certificates blade as below

Root CA Certificate

Figure 82 APIM Certificates - Root CA

Client Certificate

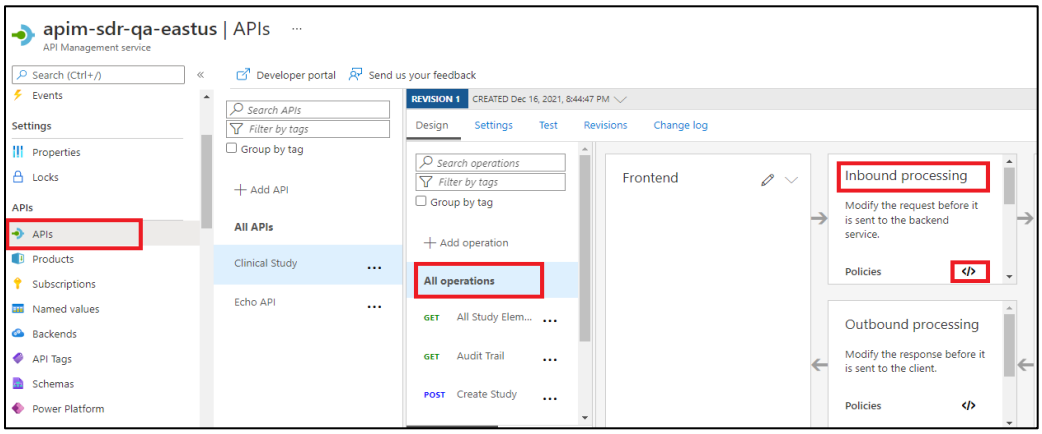
Figure 83 APIM Certificates - Client Certificate



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



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>
        <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