



Study Definition Repository (SDR) Reference Implementation

**Azure Setup and Deployment Guide
Version 3.0**

Document History

Version No.	Date	Author	Revision Description
V1.0	15-Mar-2022	Infra Team	Initial Draft
V2.0	29-Jun-2022	Mani / Jabina	Removed Smoke Testing section
V3.0	19-Aug-2022	Mani	Updated Section 2.8 manual configuration changes and Section 5 PaaS setup as per the latest infra changes.

Table of Contents

1. Introduction	6
1.1. Definitions and Acronyms.....	6
1.2. Document Scope.....	6
1.3. Out of Scope	6
1.4. Audience	7
1.5. Pre-Requisites.....	7
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.....	18
2.7. Execute GitHub Action for IaC deployment	20
2.8. Manual Configuration Changes on Azure Platform.....	21
2.8.0. PaaS Setup – OAuth 2.0 Configuration for SDR UI Application	21
2.8.1. UI App Path Mapping.....	25
2.8.2. Key Vault	26
3. Resource Validation.....	30
3.1. Low-Level Design Document	30
3.2. Virtual Network	32
3.3. Subnet	35
3.4. Delegated Subnet.....	36
3.5. Other Resources	38
4. Application Code Deployment.....	38
4.1 GitHub Secrets used in WorkFlow	38
4.2 Deploy the UI Application	39
4.3 Deploy Back-End API	42
4.4 Deployment Verification	45
5. PaaS Setup	48
5.1. PaaS Setup for APIM	48

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 Azure AD - App Registration	21
Figure 16 Azure AD - App Registration - Redirect URI.....	22
Figure 17 Azure AD - App Registration - Expose an API.....	22
Figure 18 Azure AD - App Registration - API Permission	23
Figure 19 Azure AD - App Registration - Add API Permission.....	23
Figure 20 Azure AD - App Registration - API Permission - Grant Admin Consent.....	24
Figure 21 Azure AD - App Registration - Certificates & Secrets	24
Figure 22 Azure AD - App Registration - Essential Secrets.....	25
Figure 23 App Service Path Mapping.....	26
Figure 24 Add Key Vault Access Policy.....	28
Figure 25 Select Secret Permissions in Access Policy in Key Vault	28
Figure 26 Select Principal (List of users) In Key Vault Access Policy	29
Figure 27 Create Secrets in Key Vault	29
Figure 28 Add Secrets values in Key Vault	30
Figure 29 Resource Naming Convention.....	31
Figure 30 SDR Resource Naming Convention	31
Figure 31 VNet Configurations	32
Figure 32 Virtual Network.....	33
Figure 33 Virtual Network - DDoS protection.....	33
Figure 34 Virtual Network - Tags.....	34
Figure 35 Virtual Network - Diagnostic Setting	34
Figure 36 Subnet Details.....	35
Figure 37 Subnet Details.....	36
Figure 38 Delegated Subnet-001 Details.....	37
Figure 39 Delegated Subnet-001 Service Endpoints Details	37
Figure 40 Delegated Subnet-002 Details.....	37
Figure 41 Delegated Subnet-002 Service Endpoints Details	38
Figure 42 TransCelerate GitHub Project	39
Figure 43 TransCelerate GitHub SDR UI Repo	40

Figure 44 SDR UI Repo - GitHub Actions.....	40
Figure 45 GitHub Actions - CI Workflow	41
Figure 46 GitHub - CI Workflow Run	41
Figure 47 GitHub CI Workflow Output	42
Figure 48 TransCelerate GitHub Project	42
Figure 49 TransCelerate GitHub SDR API Repo.....	43
Figure 50 GitHub Actions CI Worklfow	43
Figure 51 GitHub CI Workflow Run	44
Figure 52 GitHub CI Workflow Run	44
Figure 53 GitHub CI Workflow Output	45
Figure 54 Azure Portal	45
Figure 55 Azure Portal - SDR UI App Service	46
Figure 56 SDR UI App Service - Advanced Tools	46
Figure 57 SDR UI App Service Advanced Tools - Go.....	47
Figure 58 Azure Kudu Tool	47
Figure 59 SDR UI Deployed Files	48
Figure 60 Client Certificate.....	49
Figure 61 APIM Certificates	49
Figure 62 APIM Certificates - Client Certificates.....	50
Figure 63 APIM Certificates - Client Certificate	50
Figure 64 APIM Inbound Processing.....	51
Figure 65 APIM - Inbound Policy.....	53

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 User Interface and Application Programming Interface.

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. 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

1.2. 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.

1.3. 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.

1.4. Audience

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

1.5. Pre-Requisites

- Access to the 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](#)

2. Azure Infrastructure

2.1. Resource Provider Registration

GOAL:

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

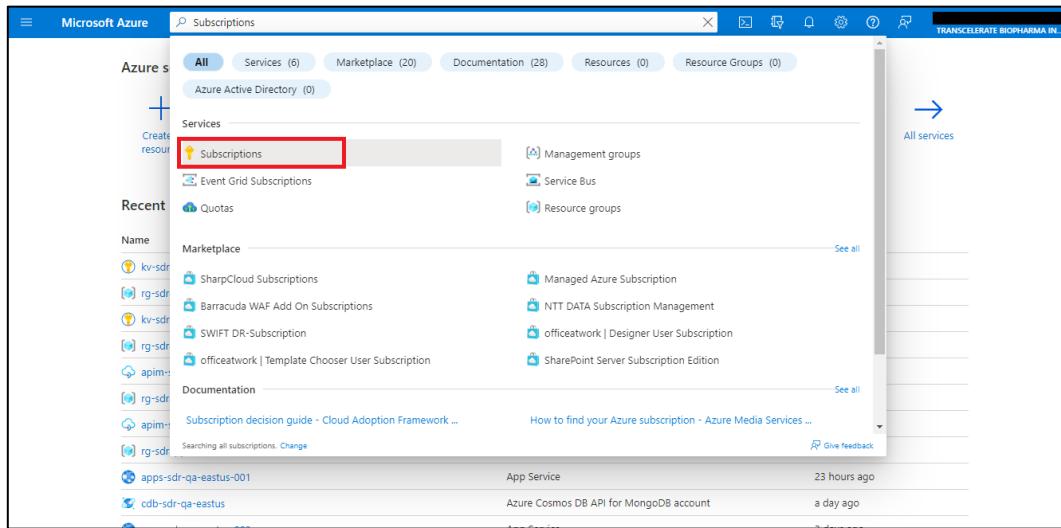
PRE-REQUISITES:

Global Admin or Subscription Owner level of access to Azure.

STEPS:

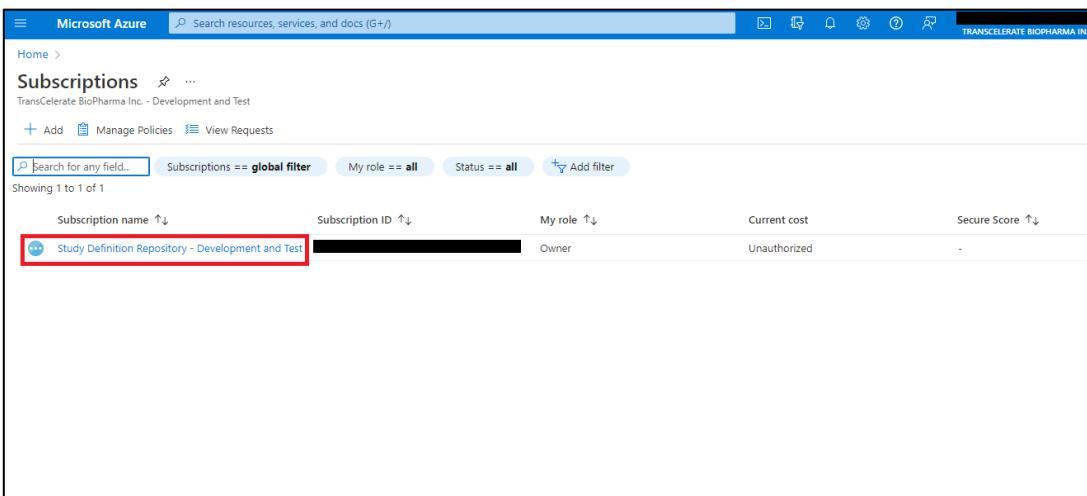
- i. Sign into the Azure portal.
- ii. On the Azure portal menu
 - Search for Subscriptions.
 - Select it from the available options as shown below.

Figure 1 Select Subscriptions in Azure Portal



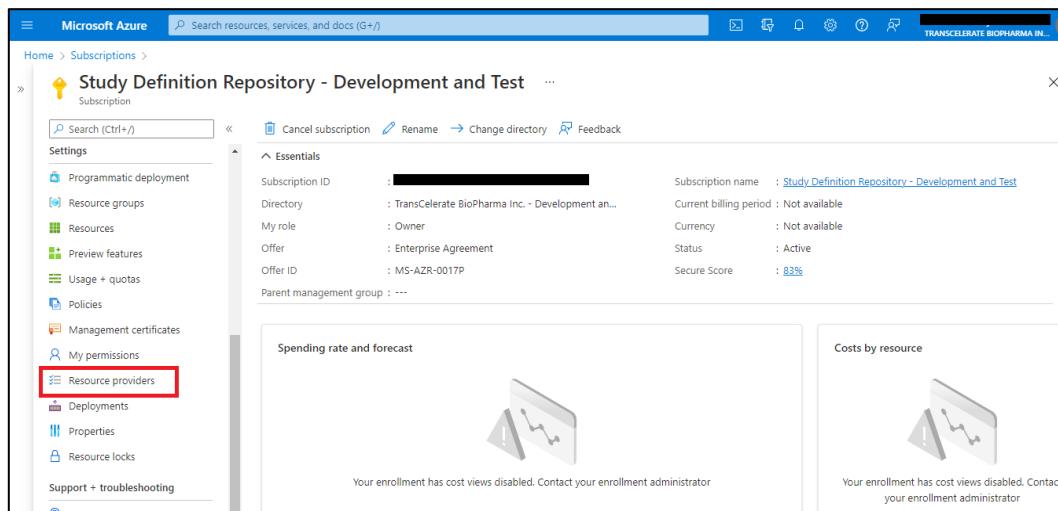
- iii. Select the subscription you want to view.

Figure 2 Select Subscription



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

Figure 3 Resource Providers in a Subscription



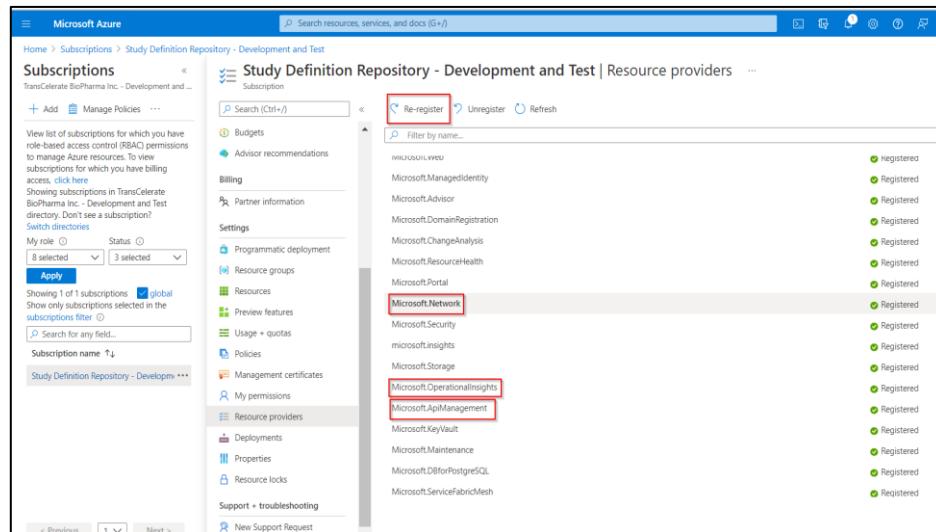
The screenshot shows the Microsoft Azure Subscriptions page for the 'Study Definition Repository - Development and Test' subscription. The left sidebar has a red box around the 'Resource providers' link under the 'Settings' category. The main content area displays subscription details like 'Subscription ID', 'Directory', 'My role', etc., and two cards: 'Spending rate and forecast' and 'Costs by resource', both with a note about cost views being disabled.

- 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' section of the Microsoft Azure Subscriptions page. A red box highlights the 'Re-register' button at the top. The list of providers includes Microsoft.Network, Microsoft.OperationalInsights, Microsoft.ApiManagement, and others, each with a status indicator showing they are 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.

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

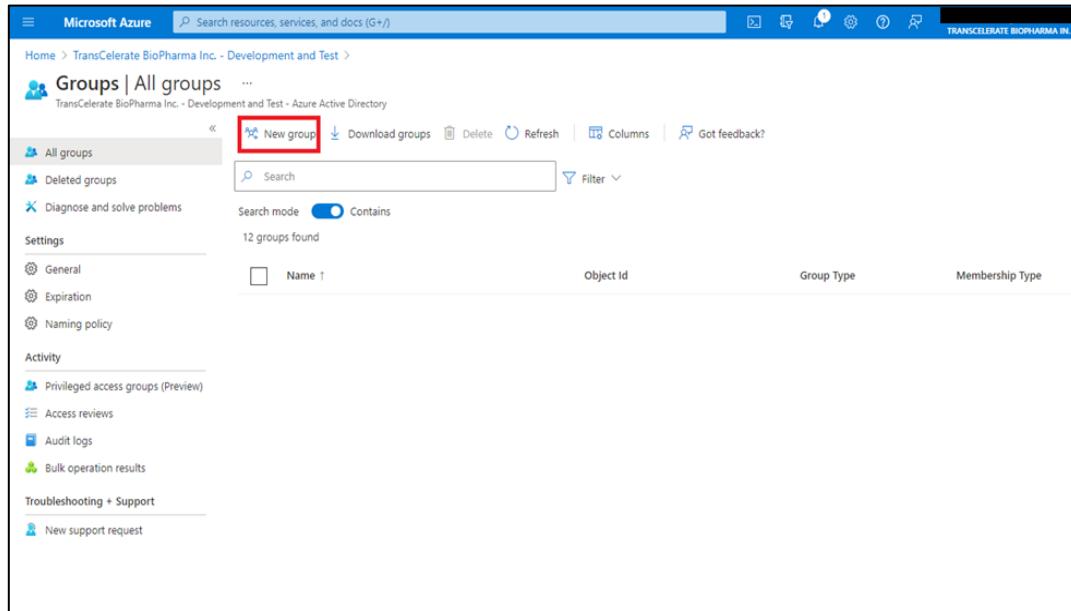
Note: To assign Azure AD roles to groups required Azure AD Premium P1 or P2 license, since this solution is part of MVP leveraged Azure AD Free plan. Follow the [Microsoft](#) Doc to assign Azure AD role to groups.

STEPS:

- i. Login to Azure portal
- ii. Search for Azure Active Directory

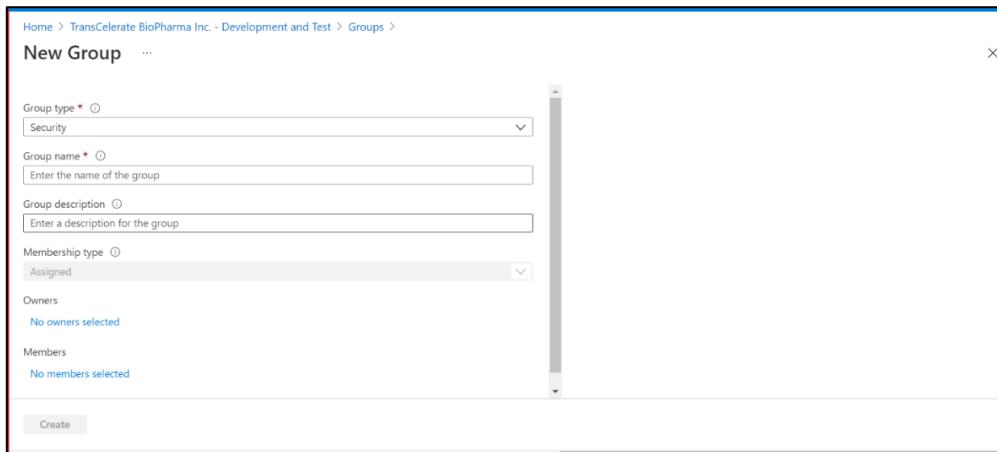
- iii. Click on the Groups on the left panel in AAD.
- iv. Click on New group tab on the top as shown below, add the security group and save the changes by adding the members to the group.

Figure 5 Adding new groups



The screenshot shows the Microsoft Azure portal interface for managing groups in Azure Active Directory. The left sidebar has 'Groups' selected. The main area shows a list of groups with a 'New group' button highlighted by a red box. The button has a plus sign icon and the text 'New group'. Other buttons in the header include 'Download groups', 'Delete', 'Refresh', 'Columns', and 'Got feedback?'. Below the header is a search bar and a 'Filter' dropdown. The table lists 12 groups found, with columns for 'Name', 'Object Id', 'Group Type', and 'Membership Type'. The 'Name' column is currently sorted in ascending order.

Figure 6 Add New Group



The screenshot shows the 'New Group' dialog box. It includes fields for 'Group type' (set to 'Security'), 'Group name' (empty), 'Group description' (empty), 'Membership type' (set to 'Assigned'), and sections for 'Owners' (empty) and 'Members' (empty). At the bottom is a 'Create' button.

2.3. Create Users

Goal:

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

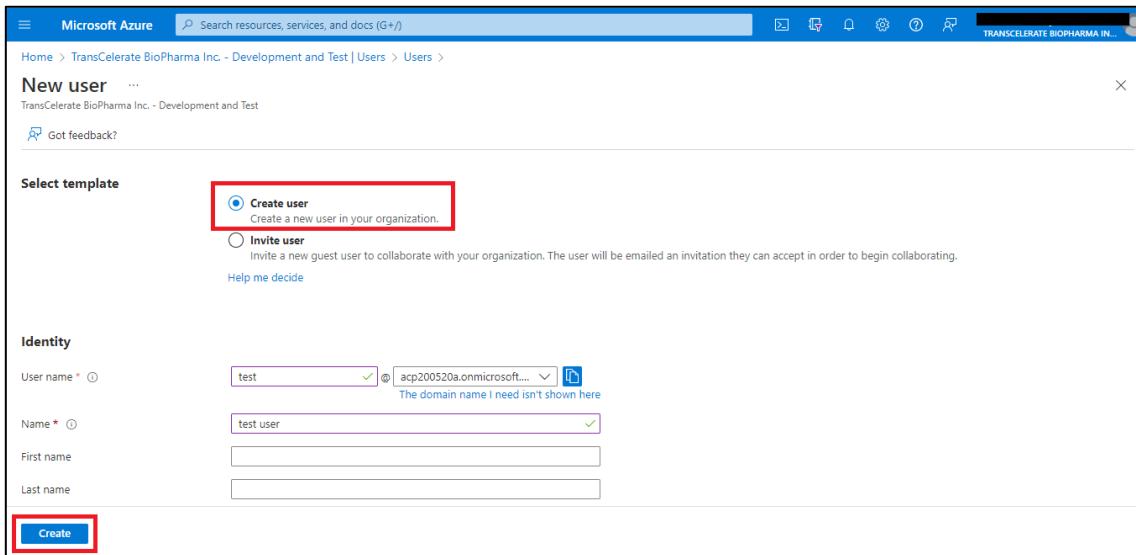
PRE-REQUISITES:

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

STEPS:

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

Figure 7 Create New User



2.4. Role Based Access Controls (RBAC)

GOAL:

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

PRE-REQUISITES:

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

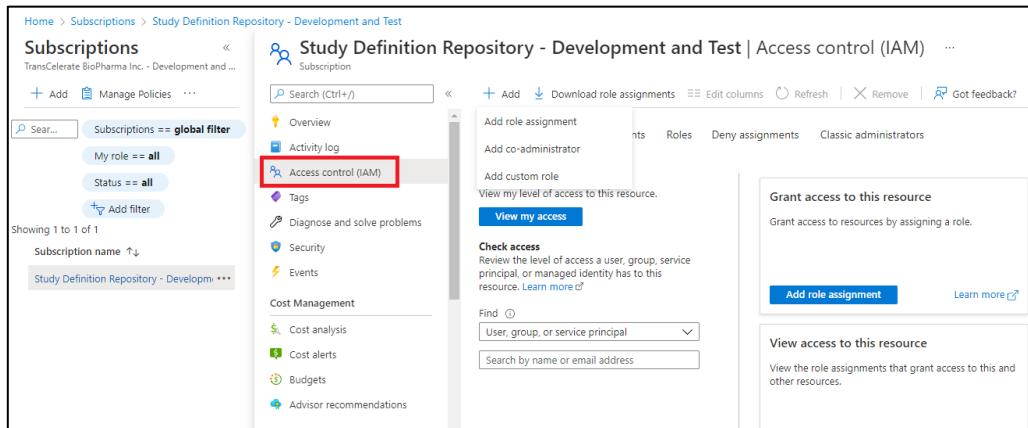
STEPS:

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

At subscription level

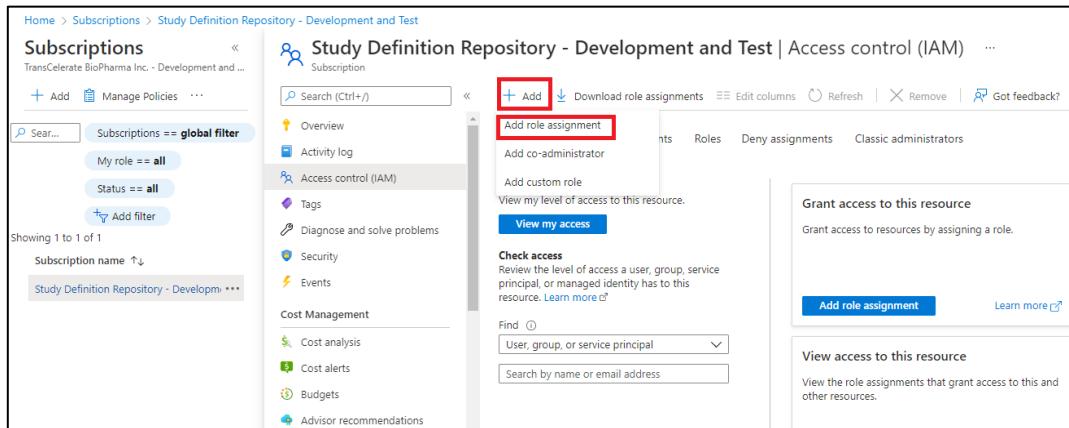
- i. Go to the subscription.
- ii. On the left pane select Access control (IAM) as shown in below screenshot.

Figure 8 Access Control (IAM)



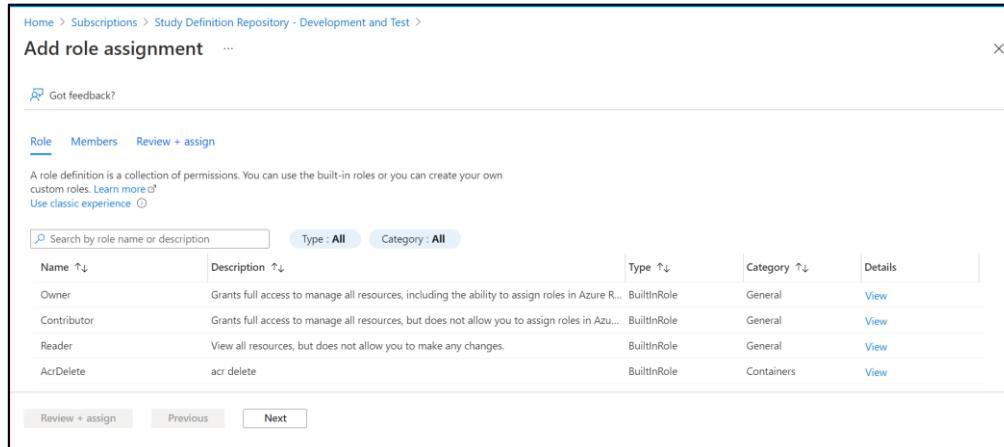
iii. Click on + Add and add the role assignment

Figure 9 Add Role Assignment



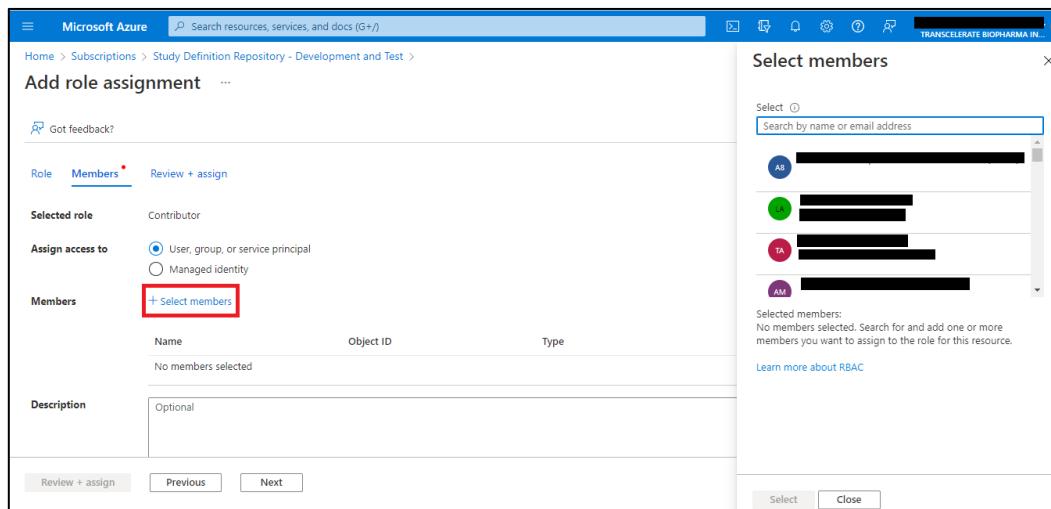
iv. Select the required role (ex: reader, contributor etc., Refer Table 1) for the members and assign the role to the created groups as shown in the screenshot below and save the changes.

Figure 10 Select Roles



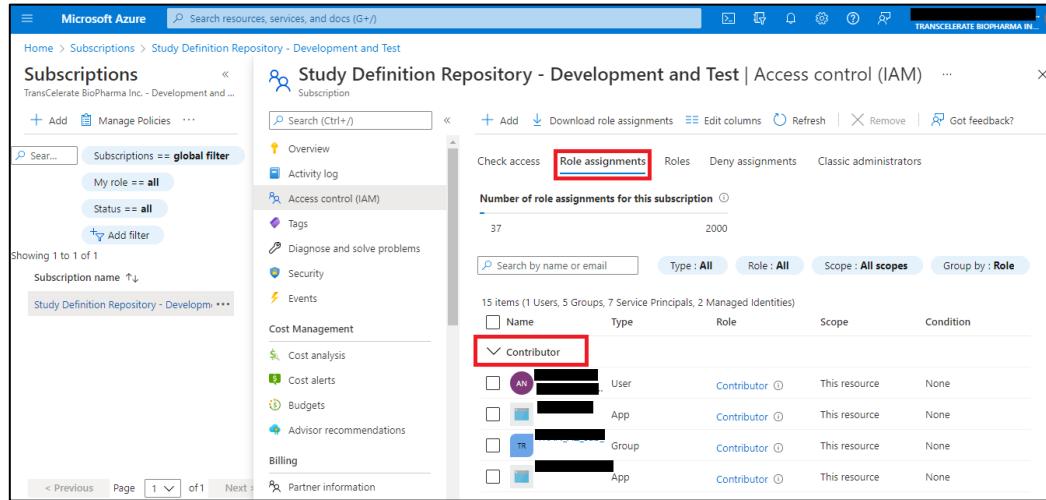
Name	Description	Type	Category	Details
Owner	Grants full access to manage all resources, including the ability to assign roles in Azure R...	BuiltInRole	General	View
Contributor	Grants full access to manage all resources, but does not allow you to assign roles in Aze...	BuiltInRole	General	View
Reader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

Figure 12 View Role Assignments



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

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

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

2.5. Storage Account and Service Principal Configuration in Azure

GOAL:

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

PRE-REQUISITES:

- Contributor level of access at Active Directory Level.

STORING THE TERRAFORM STATE FILE REMOTELY:

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

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

Table 2 Azure CLI Code Snippet - Create Storage Account

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

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

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

AZURE SERVICE PRINCIPAL:

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

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

Table 3 Azure CLI Code Snippet - Create Azure Service Principal

```
# Create Service Principal

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

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

2.6. Adding Secrets in GitHub

GOAL:

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

PRE-REQUISITES:

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

Table 4 GitHub Secrets

Secret Name	Values
AZURE_SP	The Service Principal details in JSON format.
AZURE_RESOURCEGROUP	The name of the Resource Group that contains the storage account.
AZURE_STORAGEACCOUNT	The Storage Account name
AZURE_CONTAINERNAME	The name of the blob container wherein the Terraform State file will be stored.
AZURE_CLIENT_ID	The Client Id of the service principal.
AZURE_CLIENT_SECRET	The Client Secret value of the service principal.
AZURE_SUBSCRIPTION_ID	The Azure Subscription ID
AZURE_TENANT_ID	The Azure Tenant ID
AZURE_RMKEY	Terraform state file name for each environment. (Eg: xxxxdev.tfstate).
Env	Provide the name of the environment (for example, Dev or QA), which will be added to the resource naming convention.
VNet-IP	Provide the VNet Address Space
Subnet-IP	Provide the Subnet Address Space
Subnet-Dsaddress1	Provide the Delegated Subnet1 Address Space
Subnet-Dsaddress2	Provide the Delegated Subnet2 Address Space
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.

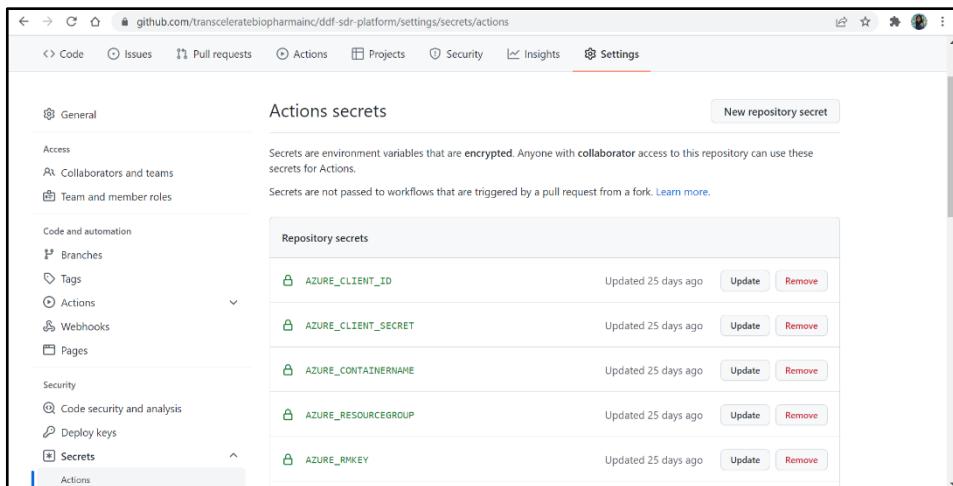
Secret Name	Values
Serviceprincipal	Provide the name of the Service principal that was created for the Git connection; it will provide key vault secret user access and access policies for secrets on Key Vault for the Service Principal.

STEPS:

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

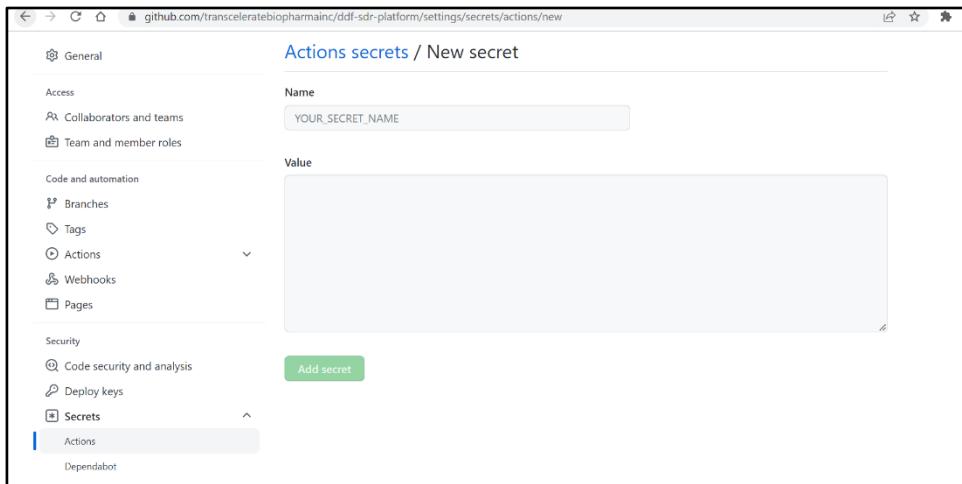
- i. Go to repository settings.
- ii. On the lower left-hand side of the screen, click on Secrets.
- iii. Under that click on Actions.
- iv. Then click on New Repository Secret.

Figure 13 GitHub Actions Secrets



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

Figure 14 Add new GitHub Action Secret



2.7. Execute GitHub Action for IaC deployment

GOAL:

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

PRE-REQUISITES:

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

DEPLOYMENT:

The folder “.github/workflows” in the IaC Repository on GitHub contains the GitHub Actions yaml script (main.yml) for deploying the Terraform IaC code on Microsoft Azure Platform.

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

2.8.0. 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:

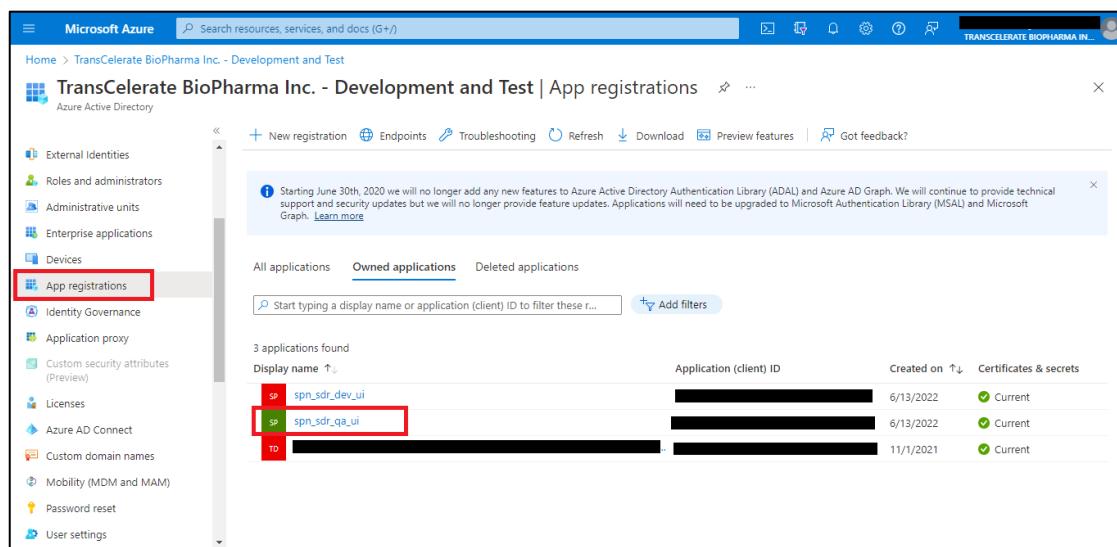
- Global Administrator level of access at Active Directory level.

STEPS

SDR UI APP REGISTRATION:

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

Figure 15 Azure AD - App Registration

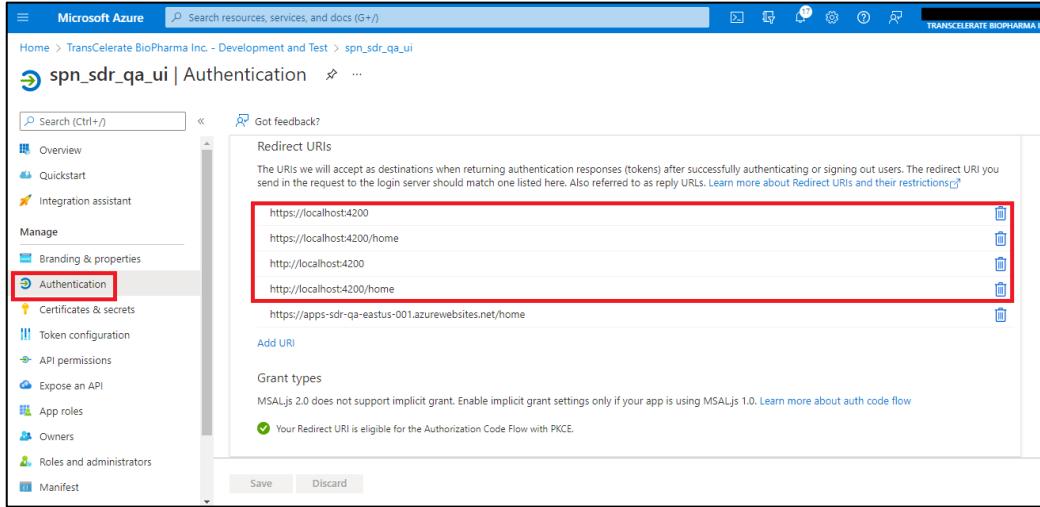


The screenshot shows the Microsoft Azure portal interface for 'App registrations'. The left sidebar has a red box around the 'App registrations' option under 'Azure Active Directory'. The main area shows a table of registered applications. Two entries are highlighted with red boxes: 'spn_sdr_dev_ui' and 'spn_sdr_qa_ui'. The table columns include 'Display name', 'Application (client) ID', 'Created on', and 'Certificates & secrets'.

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

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

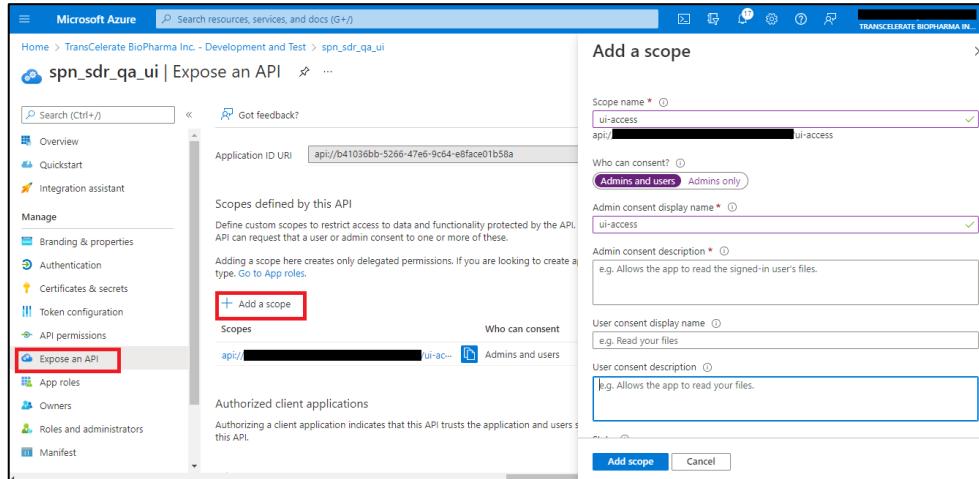
Figure 16 Azure AD - App Registration - Redirect URI



The screenshot shows the Microsoft Azure portal interface for app registration. The left sidebar navigation bar has 'Authentication' selected, indicated by a red box. The main content area is titled 'Redirect URIS'. It lists several URLs: https://localhost:4200, https://localhost:4200/home, http://localhost:4200, and http://localhost:4200/home. 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. At the bottom are 'Save' and 'Discard' buttons.

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

Figure 17 Azure AD - App Registration - Expose an API



The screenshot shows the Microsoft Azure portal interface for app registration. The left sidebar navigation bar has 'Expose an API' selected, indicated by a red box. The main content area is titled 'Add a scope'. It shows an 'Application ID URI' field with the value 'api://b41036bb-5266-47e6-9c64-e8face01b58a'. The 'Scopes defined by this API' section is visible. On the right, a modal dialog box is open with the title 'Add a scope'. It contains fields for 'Scope name' (set to 'ui-access'), 'Who can consent' (set to 'Admins and users'), 'Admin consent display name' (set to 'ui-access'), 'Admin consent description' (set to 'e.g. Allows the app to read the signed-in user's files.'), 'User consent display name' (set to 'e.g. Read your files'), and 'User consent description' (set to 'e.g. Allows the app to read your files.'). At the bottom of the dialog are 'Add scope' and 'Cancel' buttons.

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

Figure 18 Azure AD - App Registration - API Permission

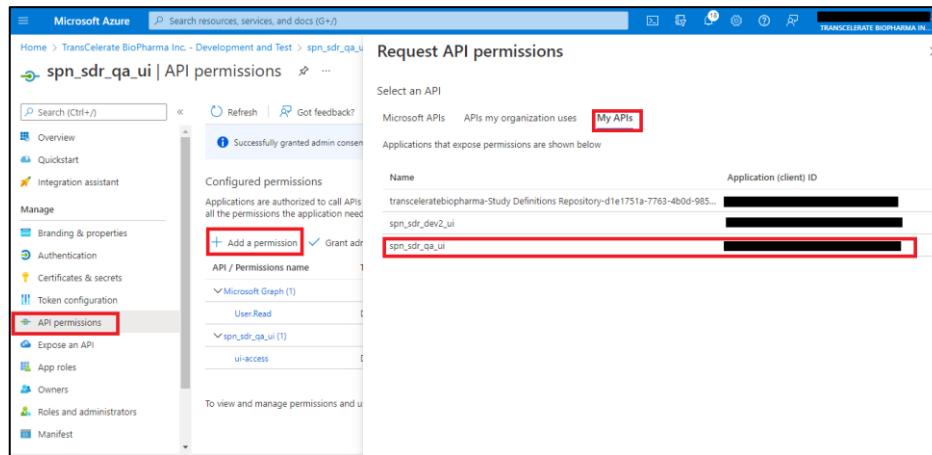
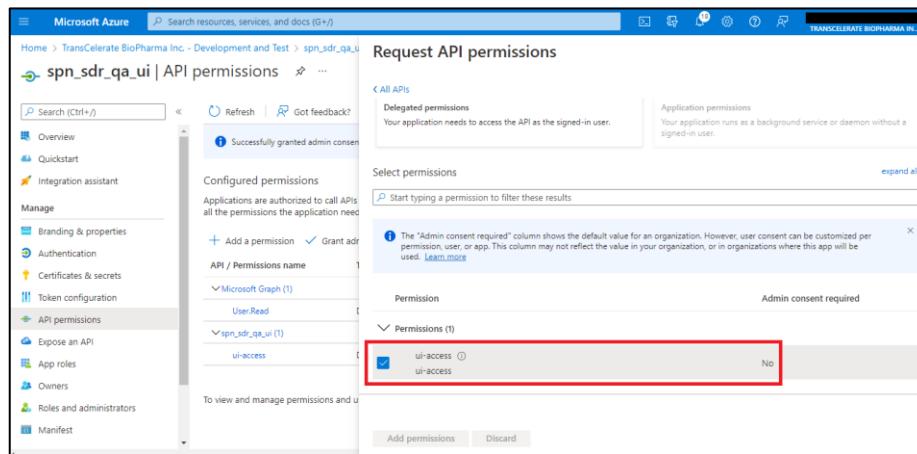
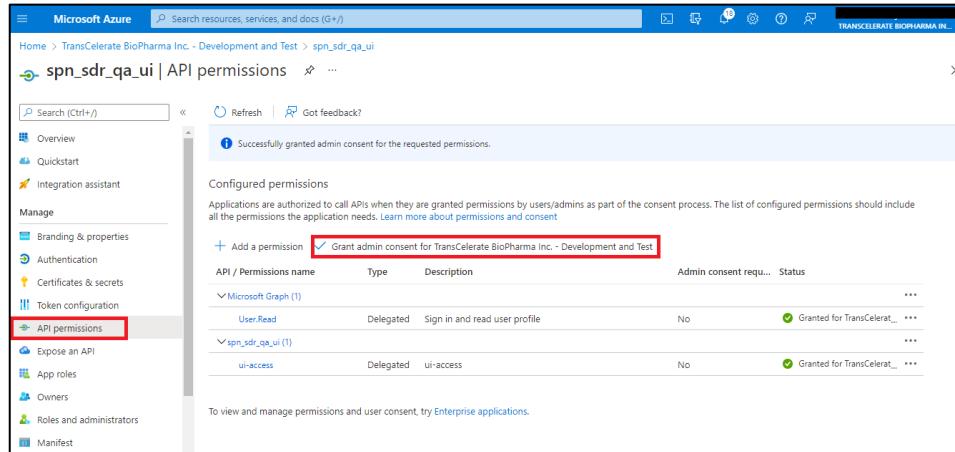


Figure 19 Azure AD - App Registration - Add API Permission



- v. Grant Admin consent.

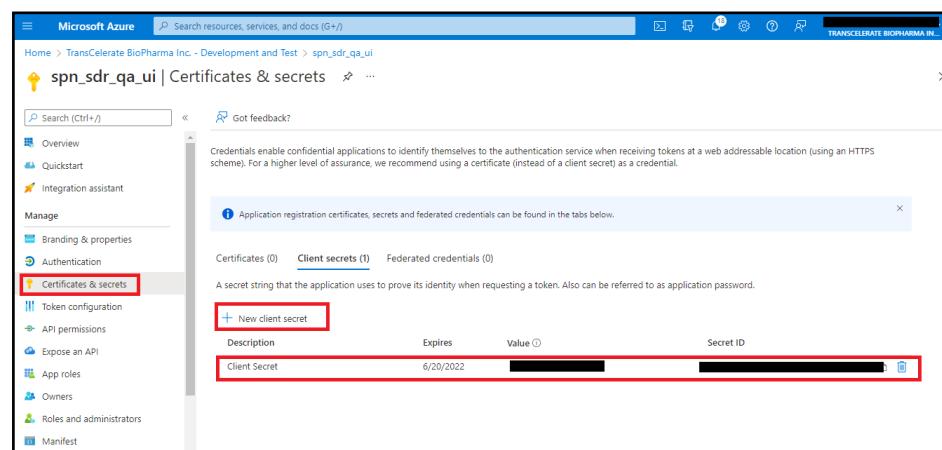
Figure 20 Azure AD - App Registration - API Permission - Grant Admin Consent



The screenshot shows the Microsoft Azure portal interface for app registration. The left sidebar lists various management options like Overview, Quickstart, Integration assistant, etc. The 'API permissions' option is highlighted with a red box. The main content area displays 'Configured permissions' for the application 'spn_sdr_qa_ui'. It shows two permissions granted by admin: 'User.Read' and 'ui-access'. Both are listed under the 'Microsoft Graph (1)' section. The status for both is 'Granted for TransCelerate BioPharma Inc. - Development and Test'. A message at the top indicates 'Successfully granted admin consent for the requested permissions.'

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

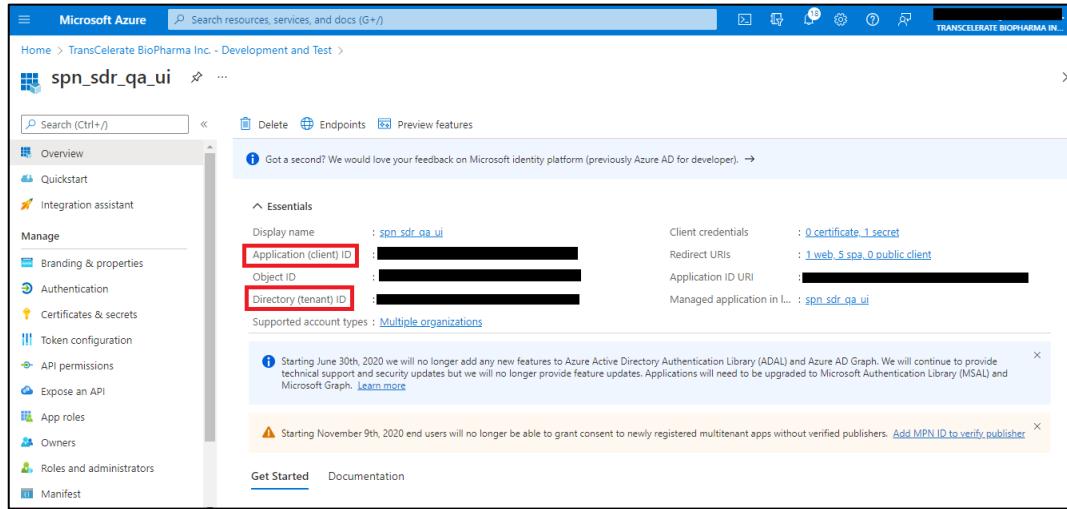
Figure 21 Azure AD - App Registration - Certificates & Secrets



The screenshot shows the Microsoft Azure portal interface for app registration. The left sidebar lists various management options like Overview, Quickstart, Integration assistant, etc. The 'Certificates & secrets' option is highlighted with a red box. The main content area displays 'Certificates (0)', 'Client secrets (1)', and 'Federated credentials (0)'. A message box says 'Application registration certificates, secrets and federated credentials can be found in the tabs below.' Below this, a table shows a single client secret entry. A red box highlights the 'Value' column, which contains the secret value 'Client Secret'. The table also includes columns for 'Description', 'Expires', and 'Secret ID'.

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

Figure 22 Azure AD - App Registration - Essential Secrets



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

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

2.8.1. UI App Path Mapping

GOAL:

Update the Path Mapping of UI Azure App Service.

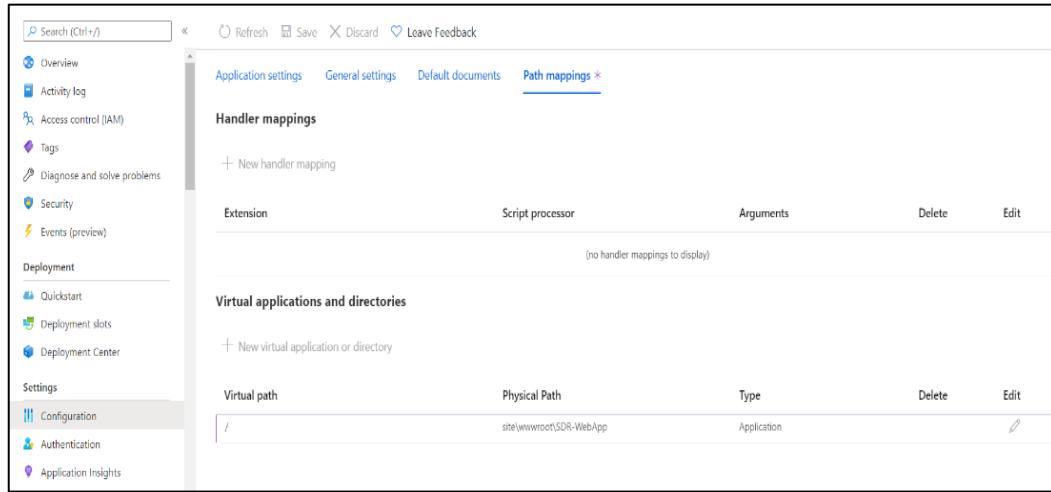
PRE-REQUISITES:

- 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.

Figure 23 App Service Path Mapping



2.8.2. Key Vault

GOAL:

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

PRE-REQUISITES:

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

Table 5 KeyVault Secrets

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

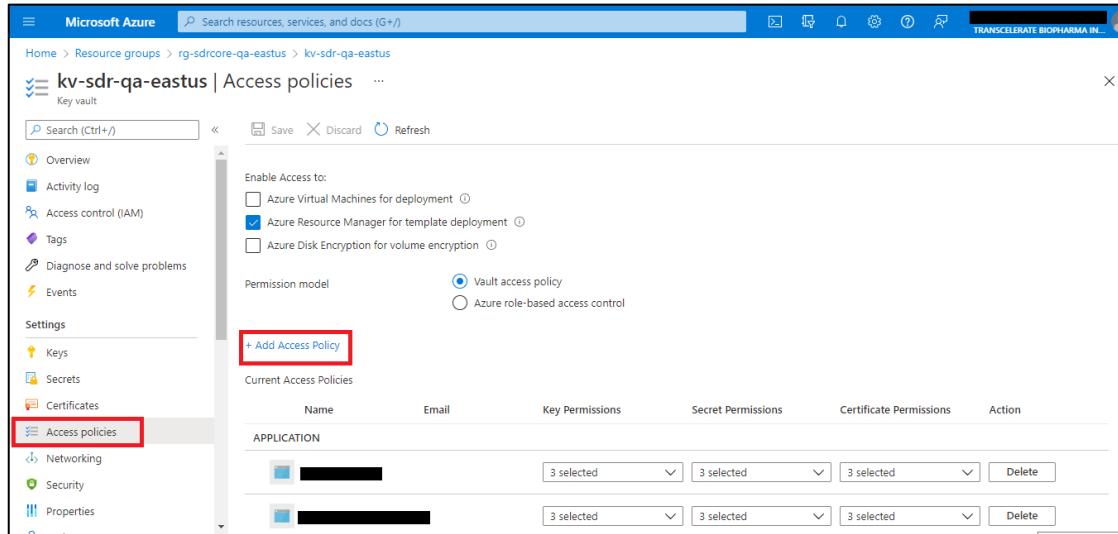
AppInsights-ApiKey	Provide Application Insights Api Key Value
AppInsights-AppId	Provide Application Insights AppId
AppInsights-RESTApiUrl	Provide Default URL https://api.applicationinsights.io/v1/apps
AzureAd-Authority	Provide the Azure Ad authority value (https://login.microsoftonline.com/ (Provide Azure AD Tenant ID))
AzureAd-ClientId	Provide the App Registration client ID, refer to Section 2.8.1 App Registration step 7 for client ID
AzureAd-LoginUrl	Provide the Front End (UI) App Service URL.
AzureAd-RedirectUrl	Provide the Redirect URL (E.g.: https://Front End App service URL (UI)/home)
AzureAd-TenantId	Provide the Azure AD Tenant ID, refer to Section 2.8.1 App Registration step 7 for Tenant ID
ConnectionStrings--DatabaseName	Provide the Cosmos DB Database Name.
ConnectionStrings--ServerName	Provide the Cosmos DB connection string.
Azure-SP	Store the Azure Service Principal JSON to utilize for API and UI deployments.
isAuthEnabled	true
isGroupFilterEnabled	true
StudyHistory--DateRange	30

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 24 Add Key Vault Access Policy

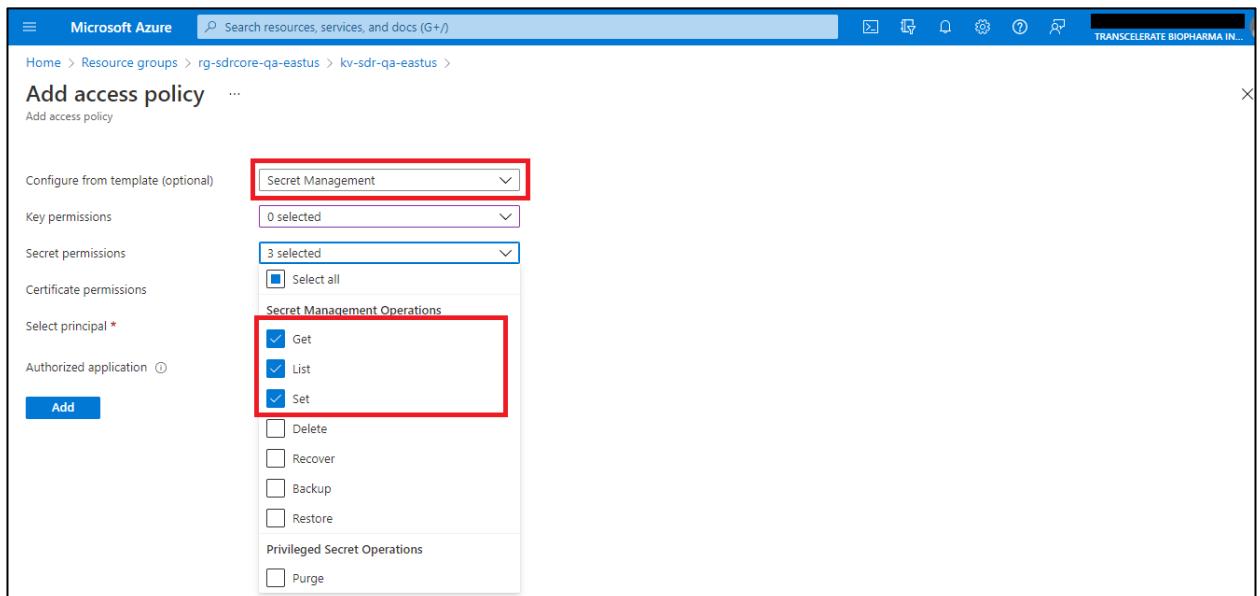


The screenshot shows the Microsoft Azure Key Vault Access Policies page. The left sidebar has 'Access policies' selected. The main area shows 'Enable Access to:' with 'Azure Resource Manager for template deployment' checked. The 'Permission model' is set to 'Vault access policy'. A red box highlights the '+ Add Access Policy' button. Below it, a table lists current access policies under 'APPLICATION'.

Name	Email	Key Permissions	Secret Permissions	Certificate Permissions	Action
[REDACTED]		3 selected	3 selected	3 selected	Delete
[REDACTED]		3 selected	3 selected	3 selected	Delete

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

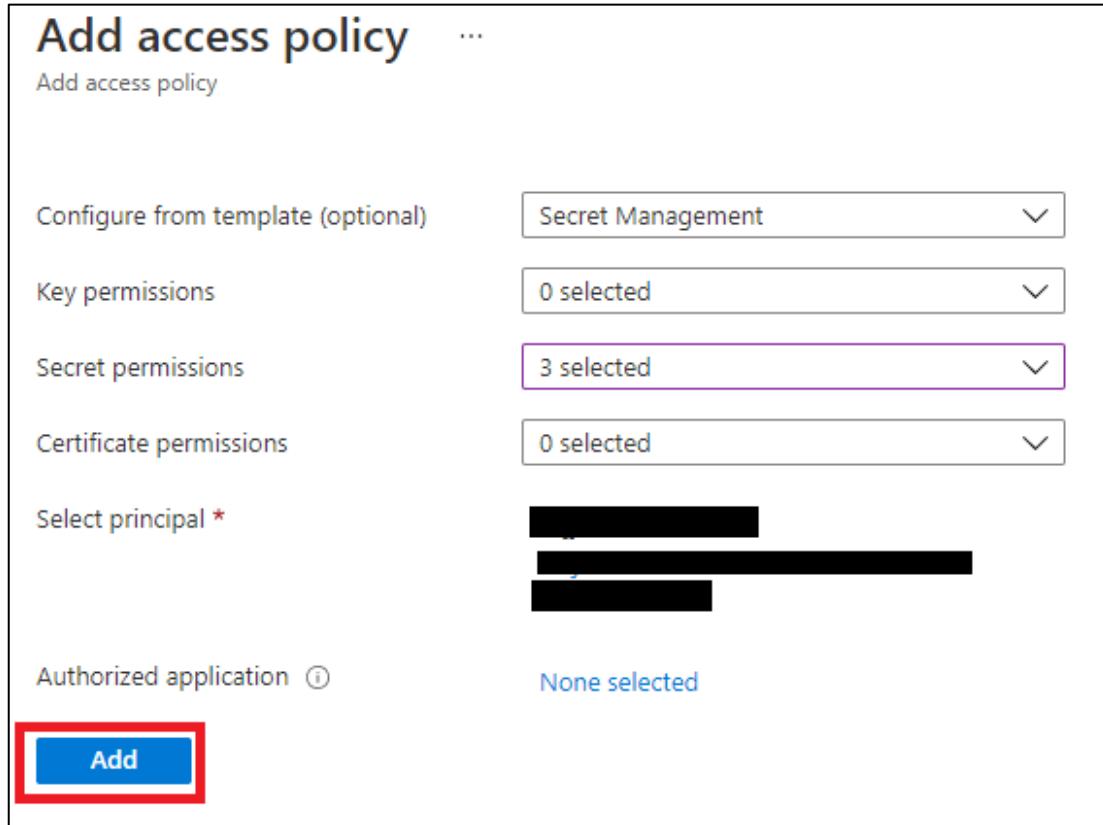
Figure 25 Select Secret Permissions in Access Policy in Key Vault



The screenshot shows the 'Add access policy' dialog. Under 'Configure from template (optional)', 'Secret Management' is selected. In the 'Secret permissions' section, 'Get', 'List', and 'Set' are checked. Other options like 'Delete', 'Recover', 'Backup', 'Restore', and 'Purge' are also listed.

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

Figure 26 Select Principal (List of users) In Key Vault Access Policy



Add access policy

Add access policy

Configure from template (optional)

Secret Management

Key permissions

0 selected

Secret permissions

3 selected

Certificate permissions

0 selected

Select principal *

Authorized application ⓘ

None selected

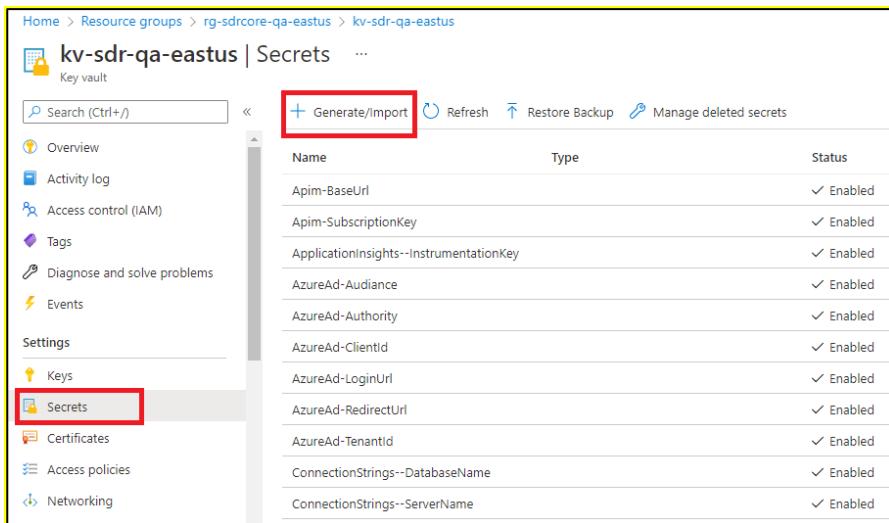
Add

STEPS FOR ADDING KEY VAULT SECRETS:

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

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

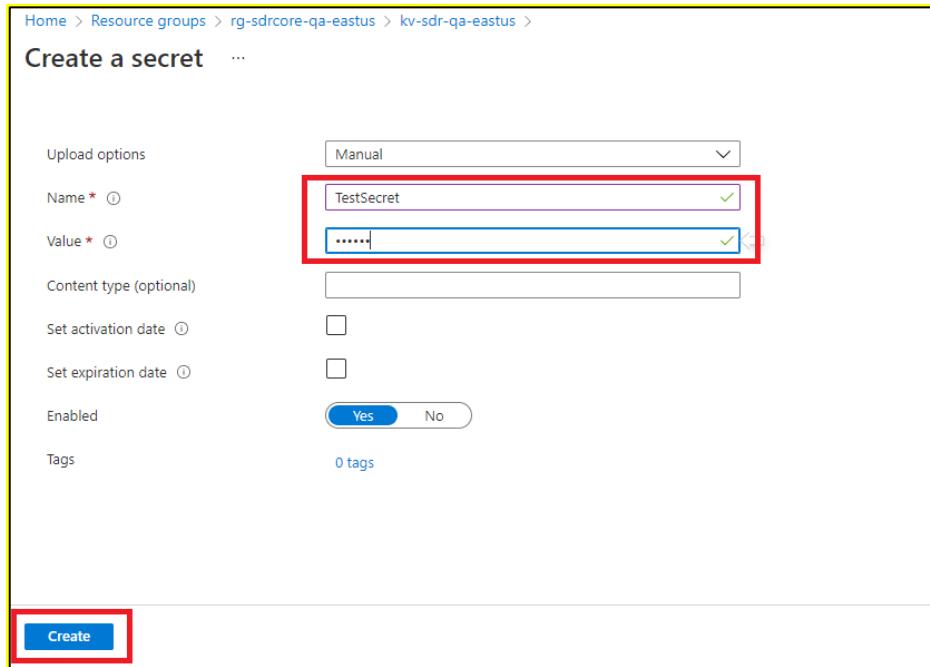
Figure 27 Create Secrets in Key Vault



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

ii. Provide Secret Name and Value → Select Create

Figure 28 Add Secrets values in Key Vault



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

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 29 Resource Naming Convention

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

Figure 30 SDR Resource Naming Convention

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

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

3.2. Virtual Network

Validation of Virtual Network (VNet) configuration

Figure 31 VNet Configurations

Vnet				
				EnvName
Basics	Project details	Subscription		Subscription Name
		Resource group		rg-SubNamecore-EnvName-eastus
IP Addresses	Instance details	Name		vnet-SubName-EnvName-eastus
		Region		East US
Security	IPv4 address space			xxx.xxx.x.x/24
Tags	BastionHost	Disable		Disable
		Enable		
Diagnostic Sett	DDoS Protection Standard	Disable		Disable
		Enable		
Diagnostic Sett	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

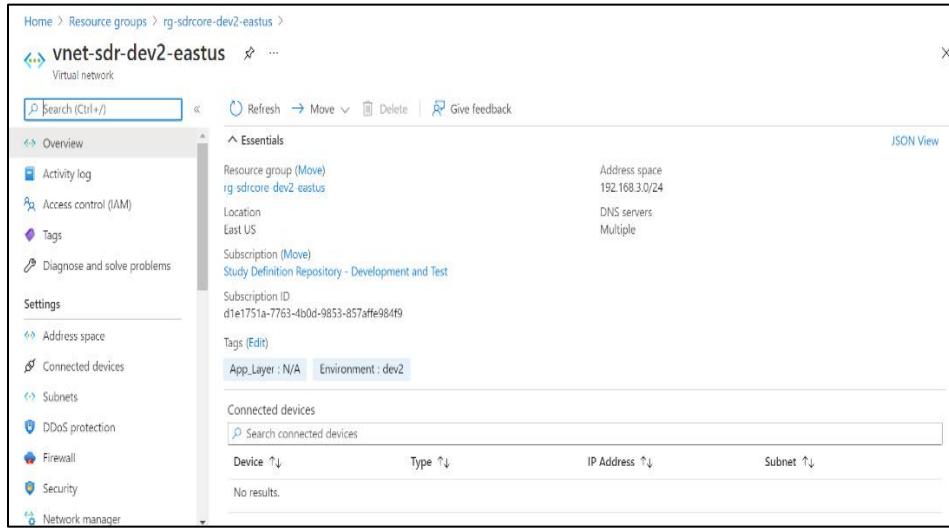
PRE-REQUISITES:

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

STEPS

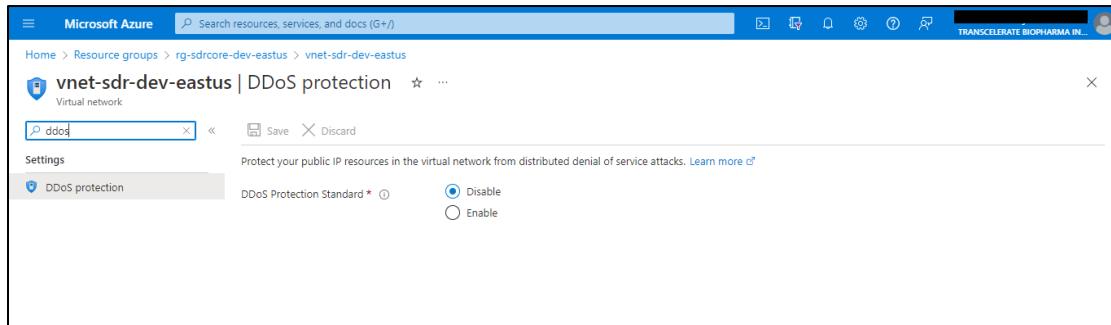
- i. Login to Azure Portal
 - ii. Click on the Resource Groups tab
 - iii. Select the Resource group for VNet configuration
 - iv. Verify that the Basic details like Subscription, Resource Group, Name and Region is as per the LLD
 - v. Verify that the IPv4 Address Space is as per the LLD

Figure 32 Virtual Network



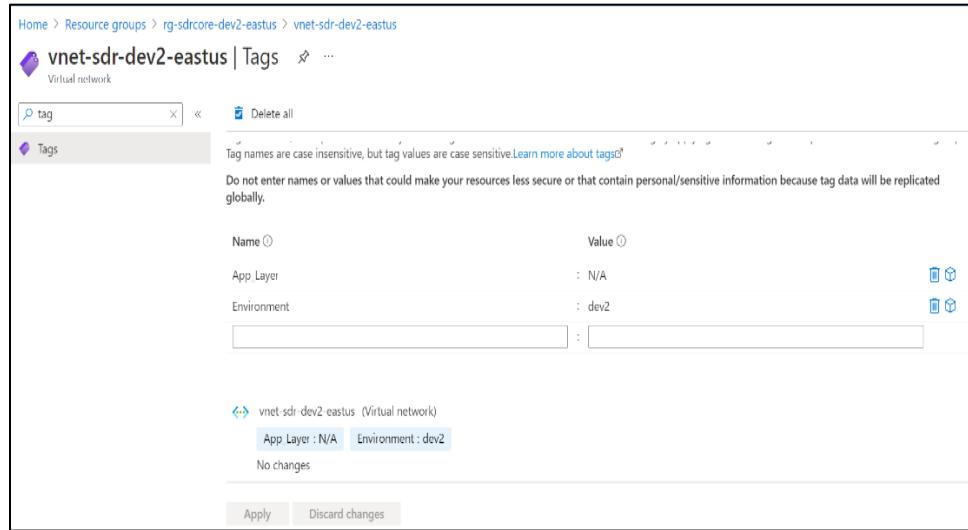
- vi. Go to Security tab and verify that Bastion Host is set as per the LLD
- vii. Go to DDoS Protection tab and verify that it is set as per the LLD

Figure 33 Virtual Network - DDoS protection



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

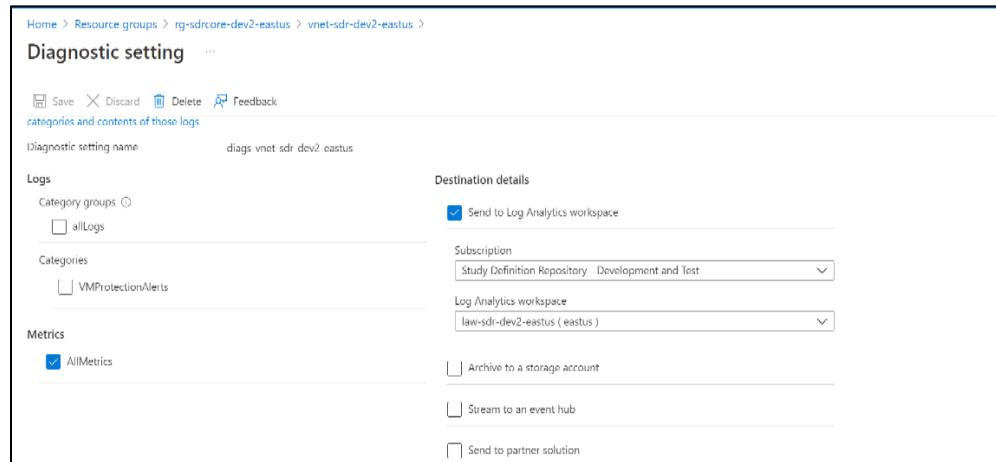
Figure 34 Virtual Network - Tags



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

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

Figure 35 Virtual Network - Diagnostic Setting



3.3. Subnet

Validation of Virtual Subnet configuration.

PRE-REQUISITES:

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

STEPS:

- 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 36 Subnet Details

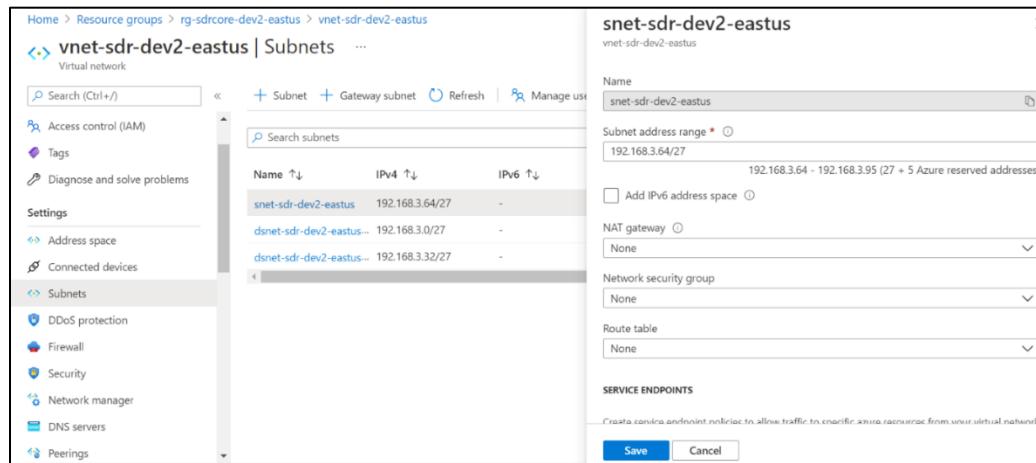
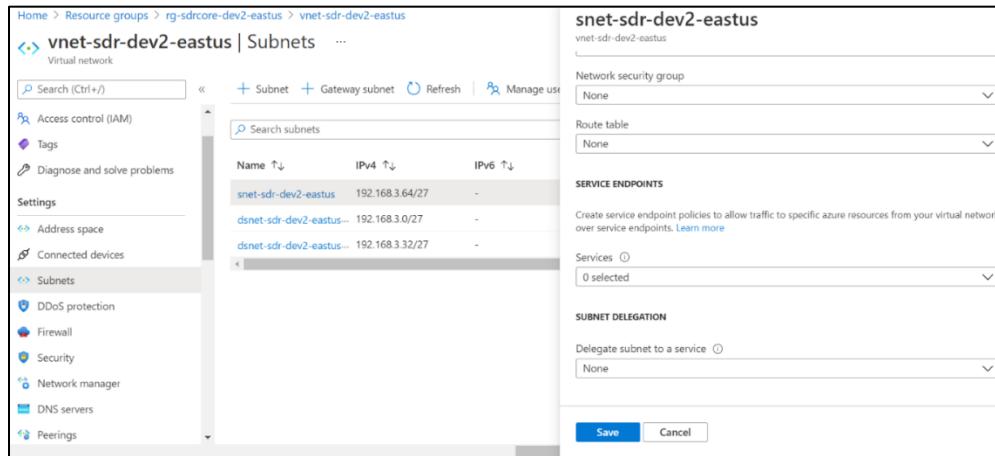


Figure 37 Subnet Details



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 38 Delegated Subnet-001 Details

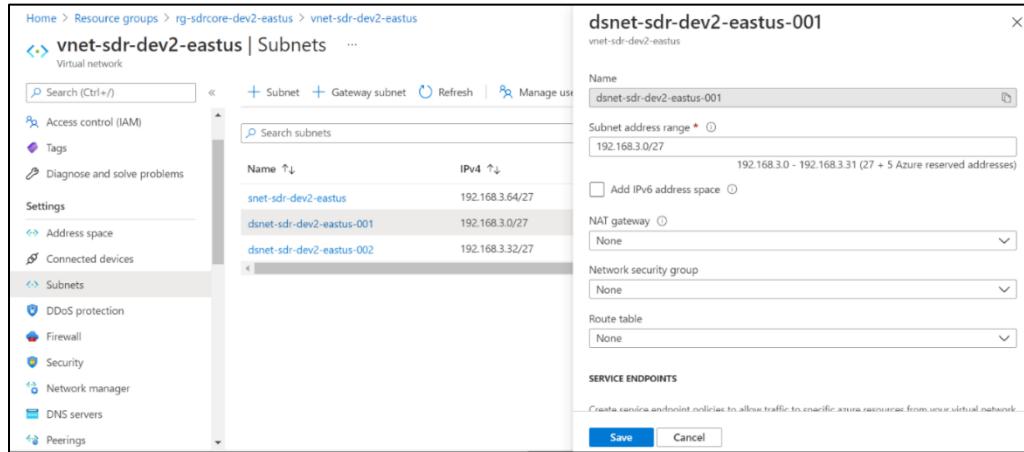


Figure 39 Delegated Subnet-001 Service Endpoints Details

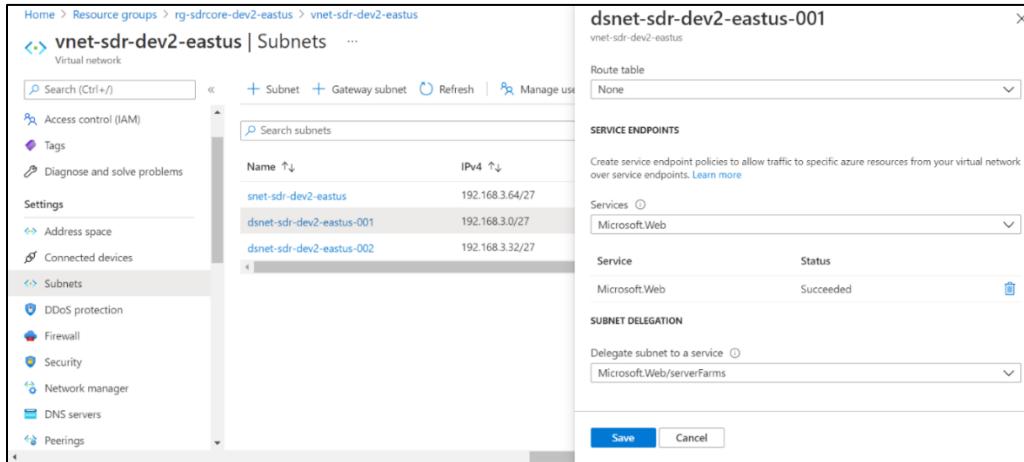


Figure 40 Delegated Subnet-002 Details

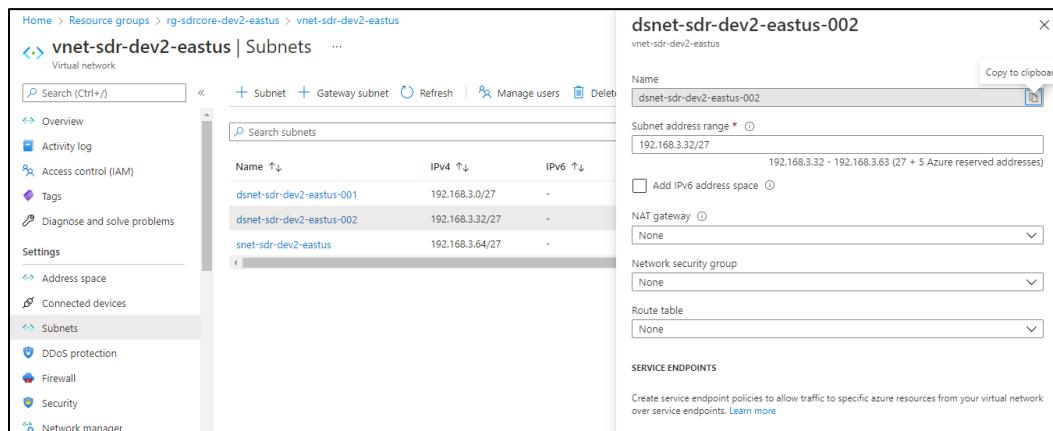
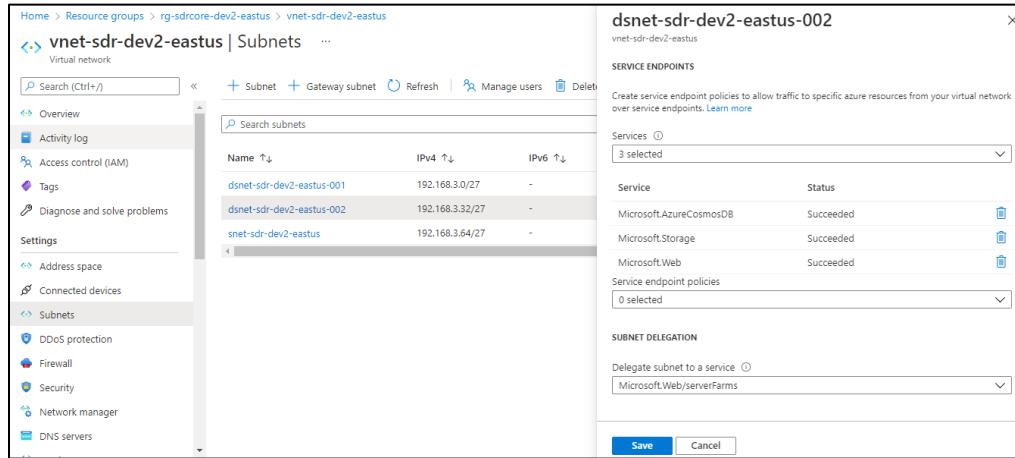


Figure 41 Delegated Subnet-002 Service Endpoints Details



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

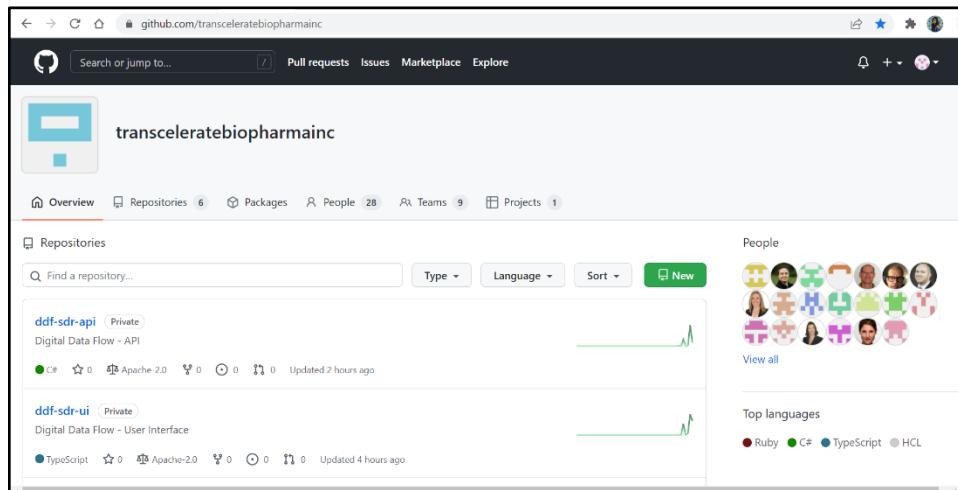
PRE-REQUISITES

- Contributor level of access at Resource Group level.

DEPLOYMENT STEPS:

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

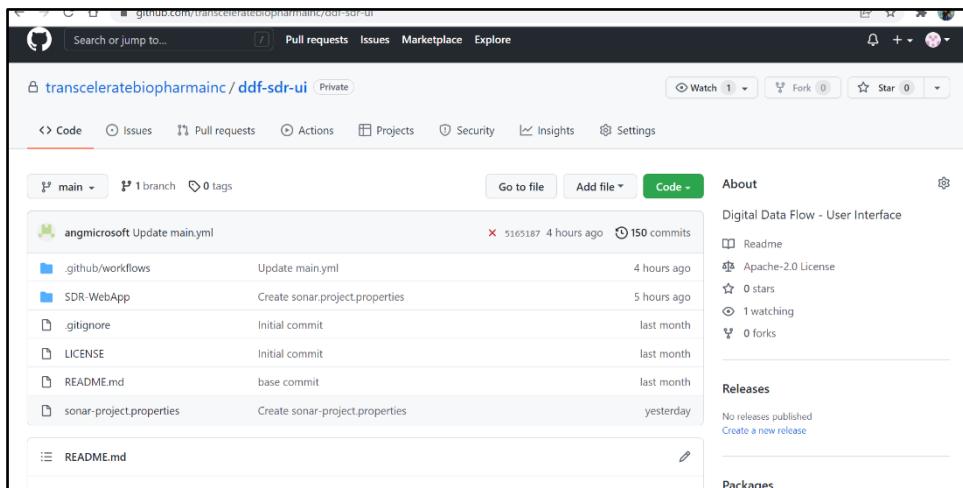
Figure 42 TransCelerate GitHub Project



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

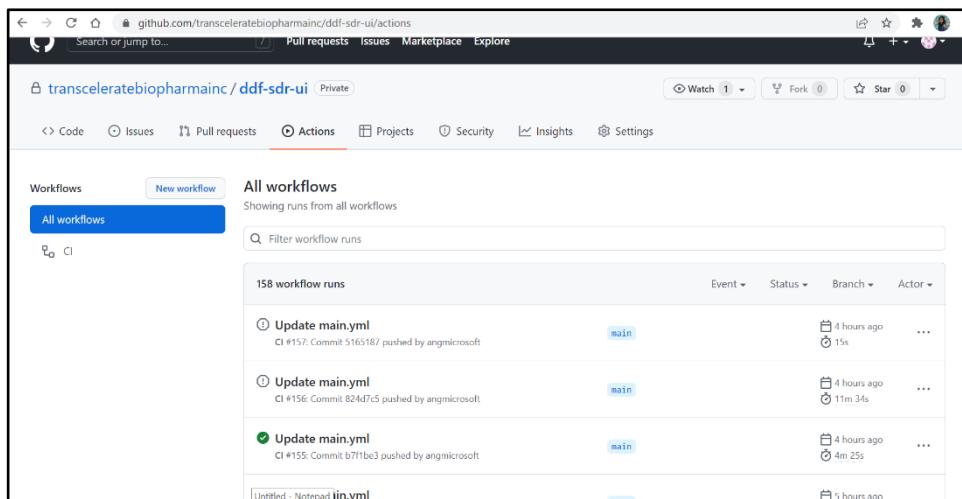
<https://github.com/transceleratebiopharmainc/ddf-sdr-ui>

Figure 43 TransCelerate GitHub SDR UI Repo



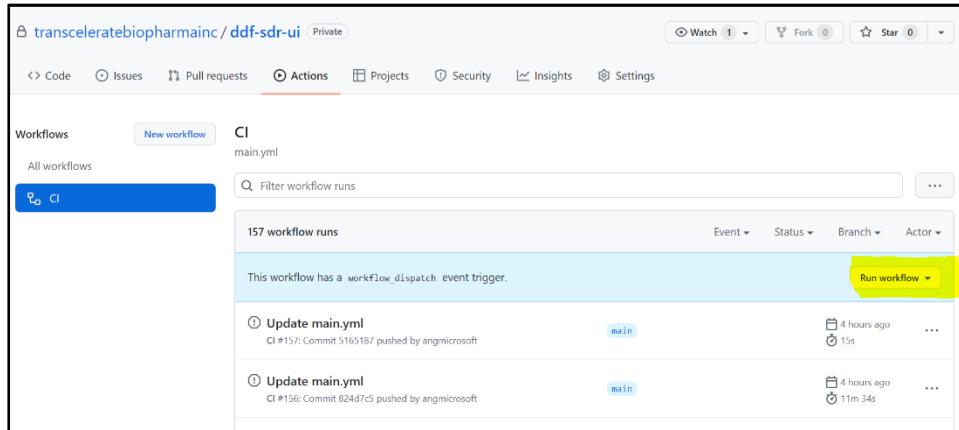
iii. Click on Actions tab.

Figure 44 SDR UI Repo - GitHub Actions



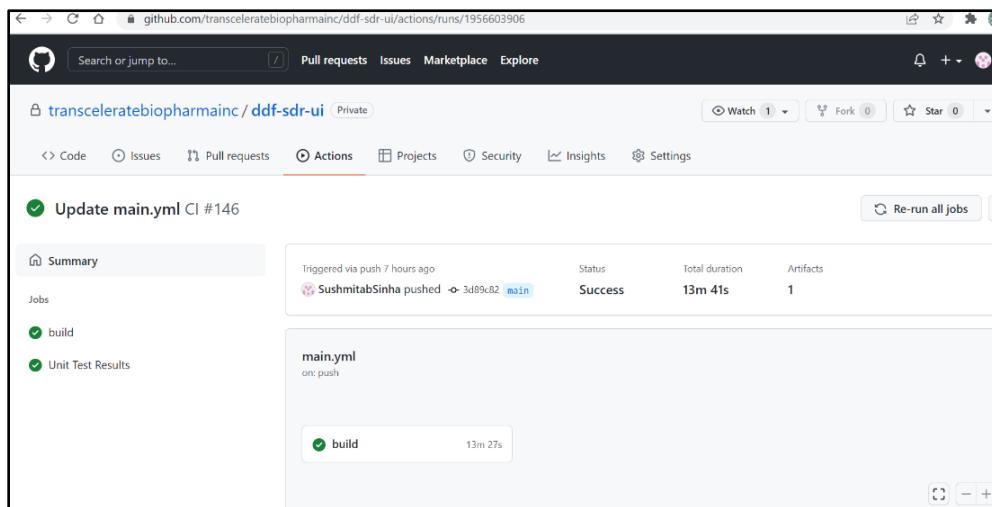
iv. Click on the workflow CI under All workflow.

Figure 45 GitHub Actions - CI Workflow



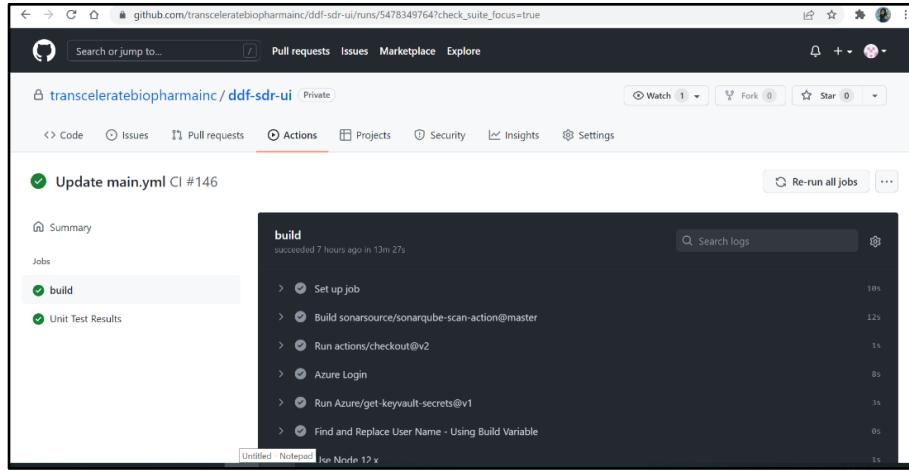
- v. Click on Run Workflow on the right-hand side. Then the action will be triggered.

Figure 46 GitHub - CI Workflow Run



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

Figure 47 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

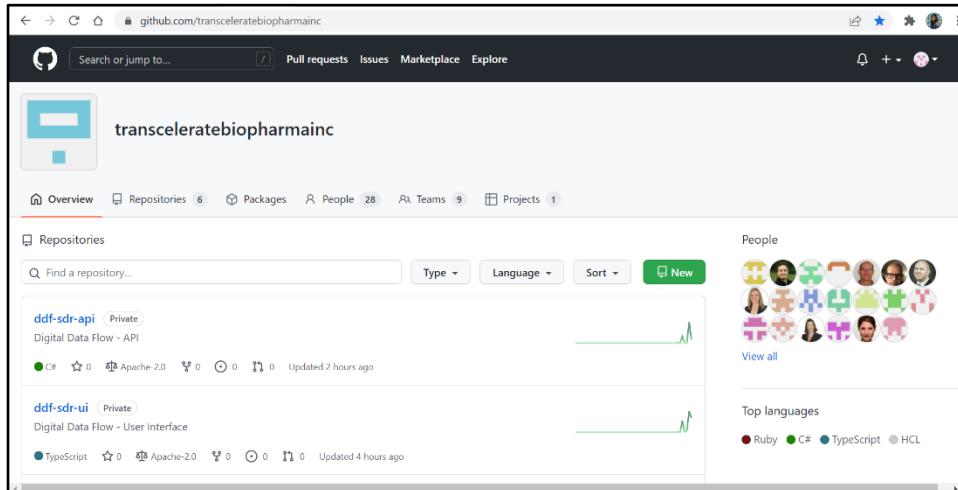
PRE-REQUISITES

- Contributor access at Resource group Level.

DEPLOYMENT STEPS:

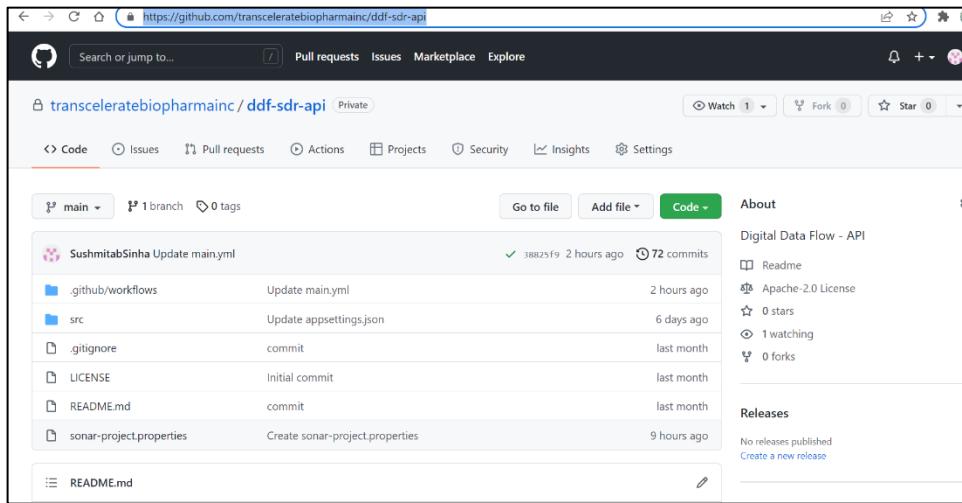
- Go to [GitHub URL](https://github.com/transceleratebiopharmainc).

Figure 48 TransCelerate GitHub Project



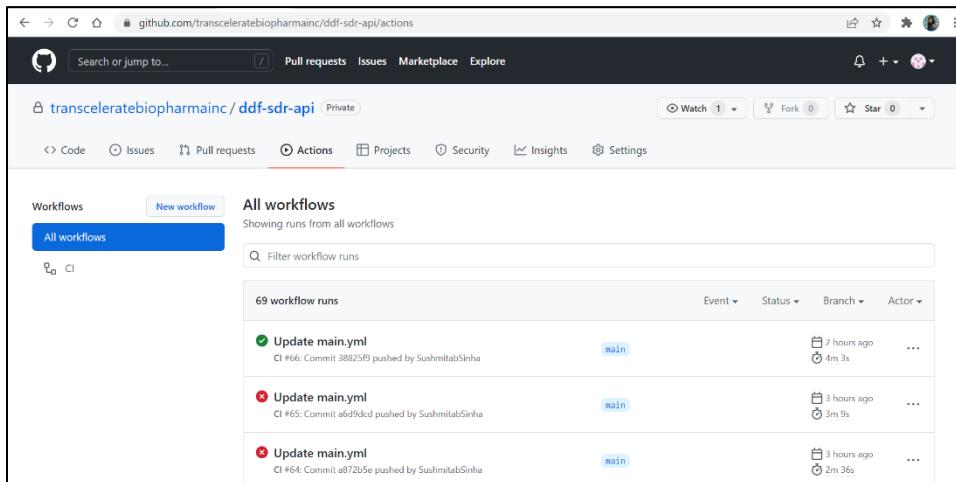
- Select the required repository, here **ddf-sdr-api**.
URL - <https://github.com/transceleratebiopharmainc/ddf-sdr-api>

Figure 49 TransCelerate GitHub SDR API Repo



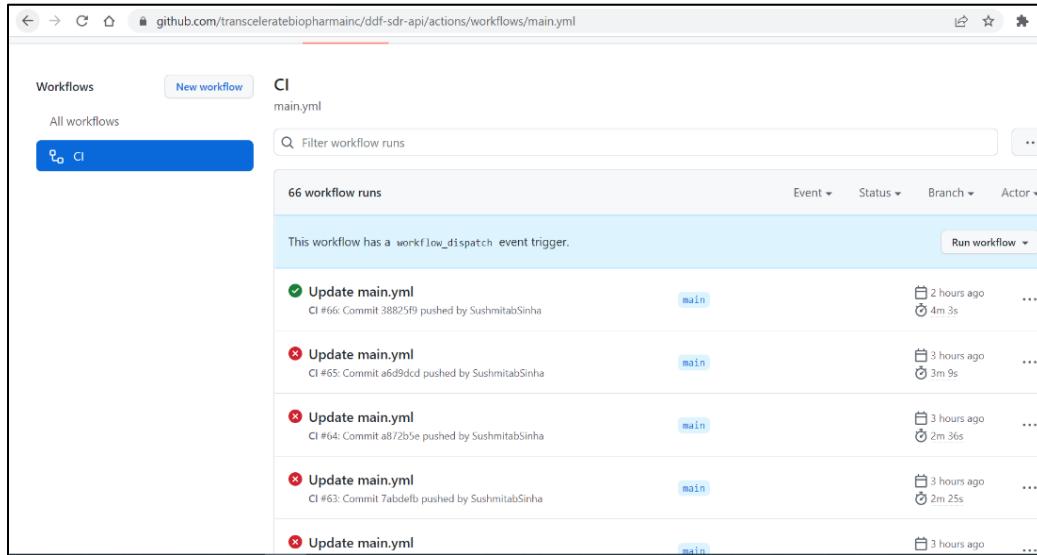
iii. Click on Actions tab.

Figure 50 GitHub Actions CI Workflow



iv. Click on the workflow CI under All workflow.

Figure 51 GitHub CI Workflow Run

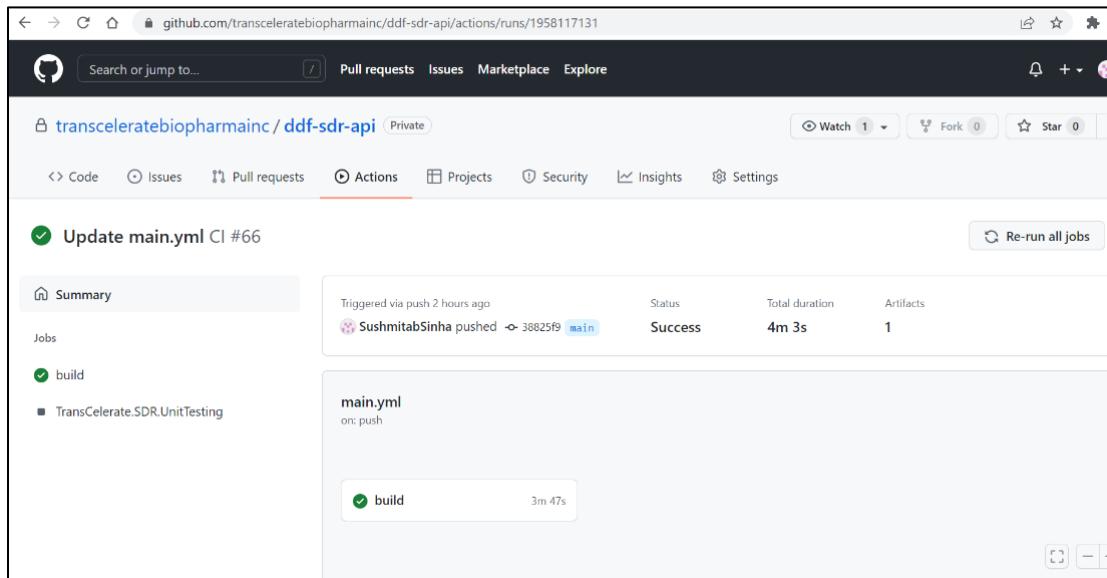


The screenshot shows the GitHub CI Workflow Runs page for the repository `ddf-sdr-api`. The main heading is `CI` and the file is `main.yml`. There are 66 workflow runs listed. The first run is successful (green checkmark) and triggered by a push event. The subsequent four runs are failing (red X). The runs are ordered from most recent at the top to oldest at the bottom. Each run row includes the status icon, commit hash, author, branch, duration, and a three-dot menu.

Run Status	Commit Hash	Author	Branch	Duration	More Options
Success	38825f9	SushmitabSinha	main	2 hours ago 4m 3s	...
Failure	a6d9dcd	SushmitabSinha	main	3 hours ago 3m 9s	...
Failure	a872b5e	SushmitabSinha	main	3 hours ago 2m 36s	...
Failure	7abdefb	SushmitabSinha	main	3 hours ago 2m 25s	...
Failure			main	3 hours ago	...

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

Figure 52 GitHub CI Workflow Run

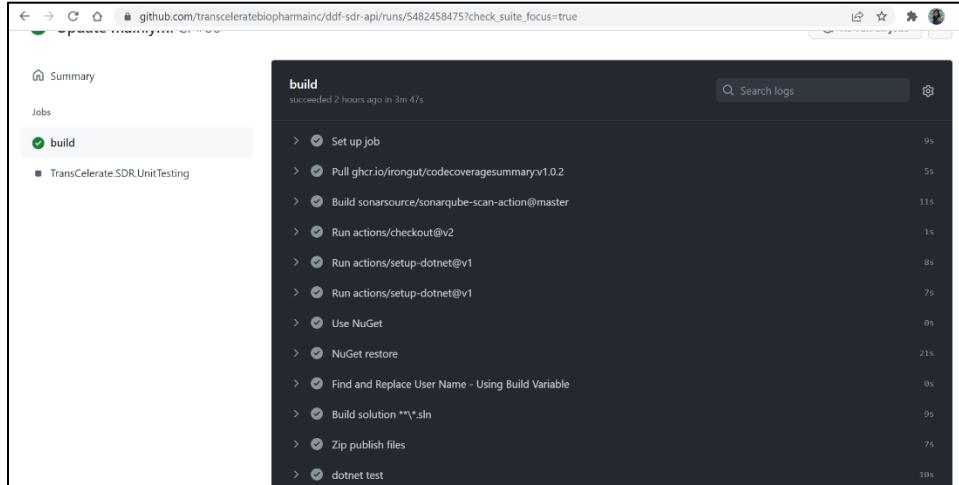


The screenshot shows the detailed view of a GitHub CI workflow run. The run is identified as `Update main.yml CI #66`. The summary table shows the run was triggered via push 2 hours ago, pushed by `SushmitabSinha` with commit `38825f9` on the `main` branch, with a success status, a total duration of `4m 3s`, and one artifact. Below the summary, the `main.yml` job is expanded, showing it was triggered on a push and contains a single `build` step that completed successfully in `3m 47s`.

Triggered via	Pushed by	Status	Total duration	Artifacts
push 2 hours ago	SushmitabSinha	Success	4m 3s	1

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

Figure 53 GitHub CI Workflow Output



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

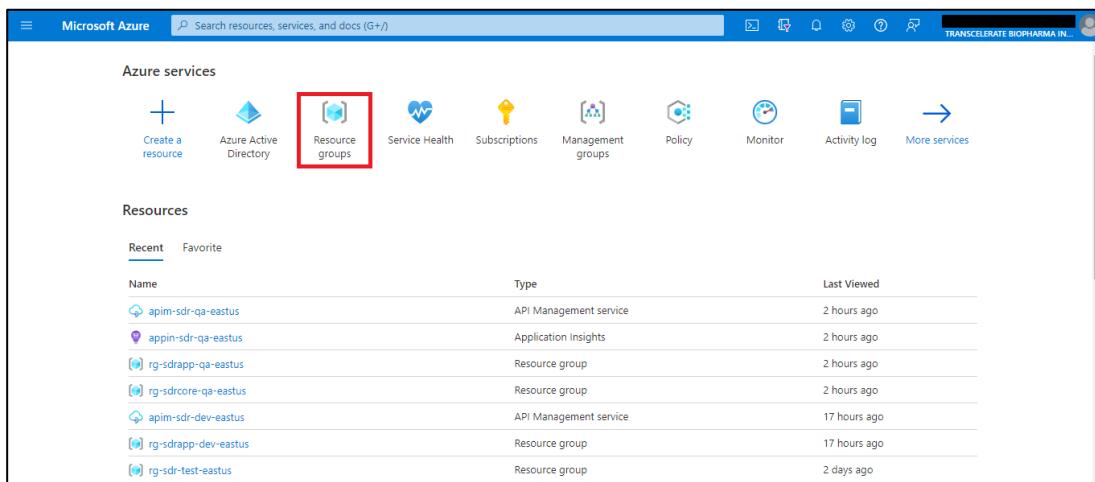
4.4 Deployment Verification

UI APPLICATION VERIFICATION STEPS:

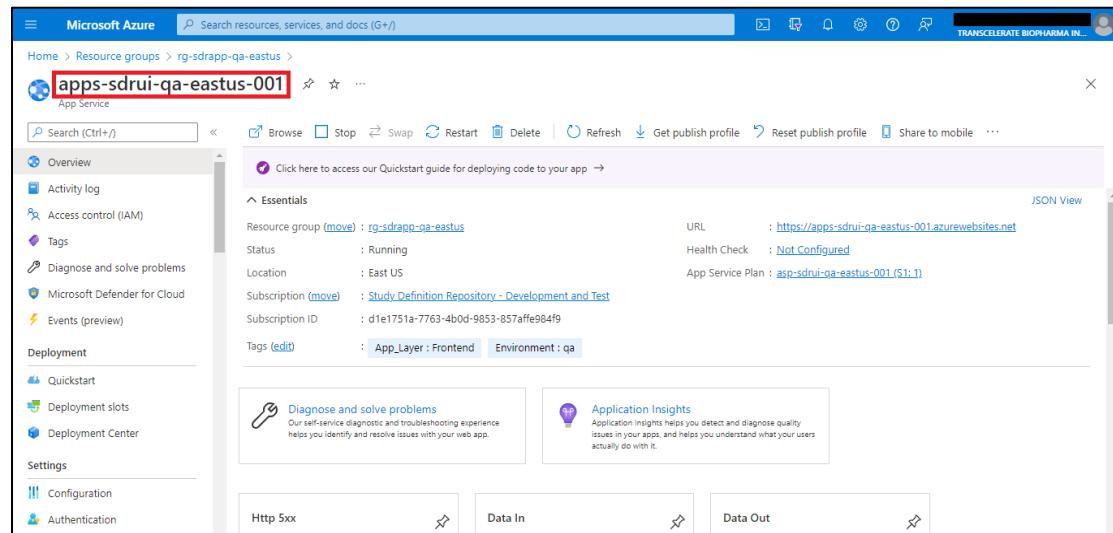
This is to verify the UI Application deployment was successful.

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

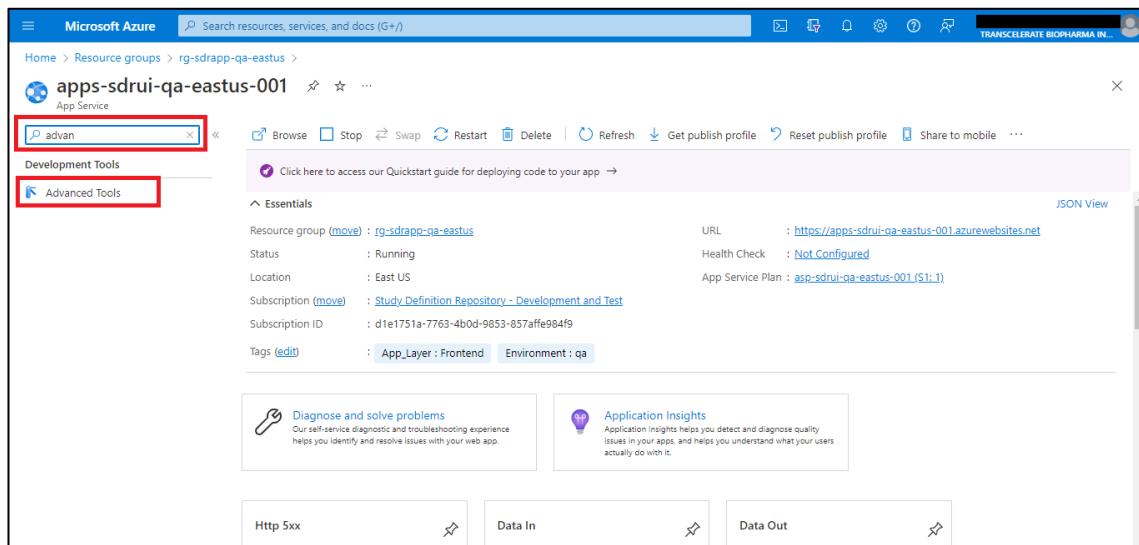
Figure 54 Azure Portal



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

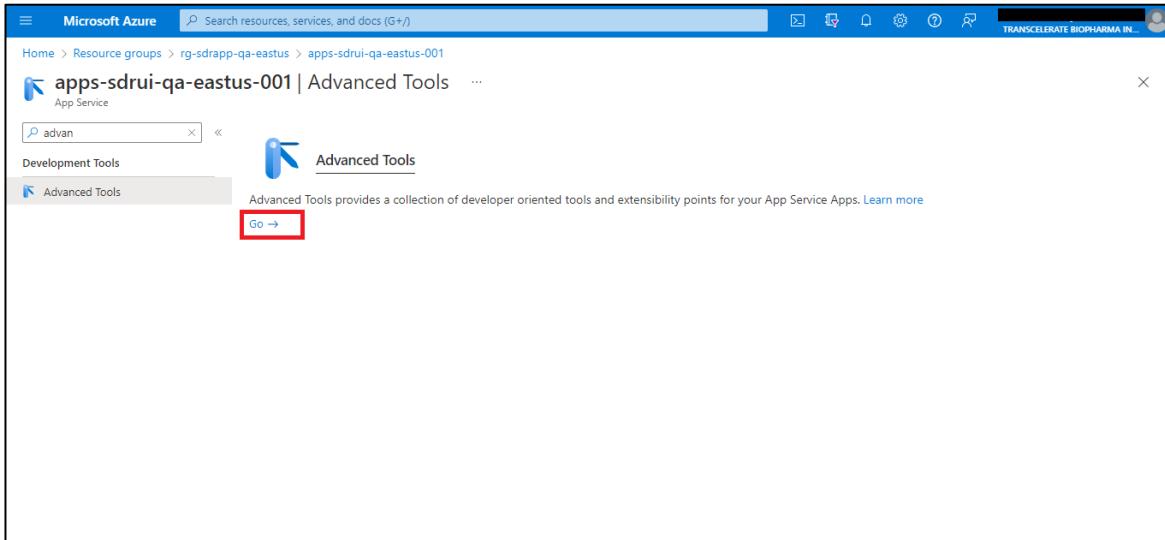
Figure 55 Azure Portal - SDR UI App Service


iii. In search box, search for Advanced tools.

Figure 56 SDR UI App Service - Advanced Tools


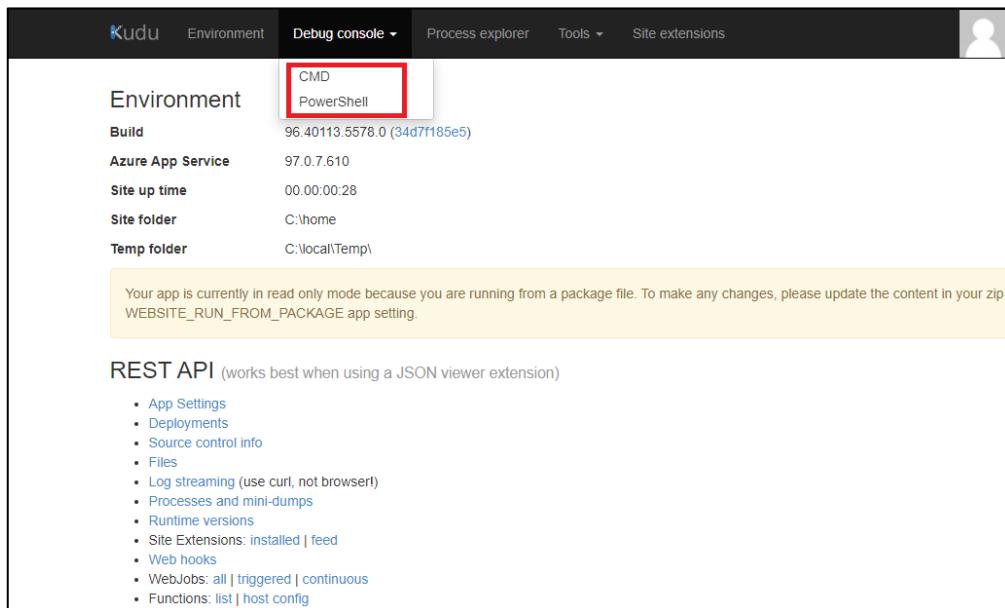
iv. Click on Go.

Figure 57 SDR UI App Service Advanced Tools - Go



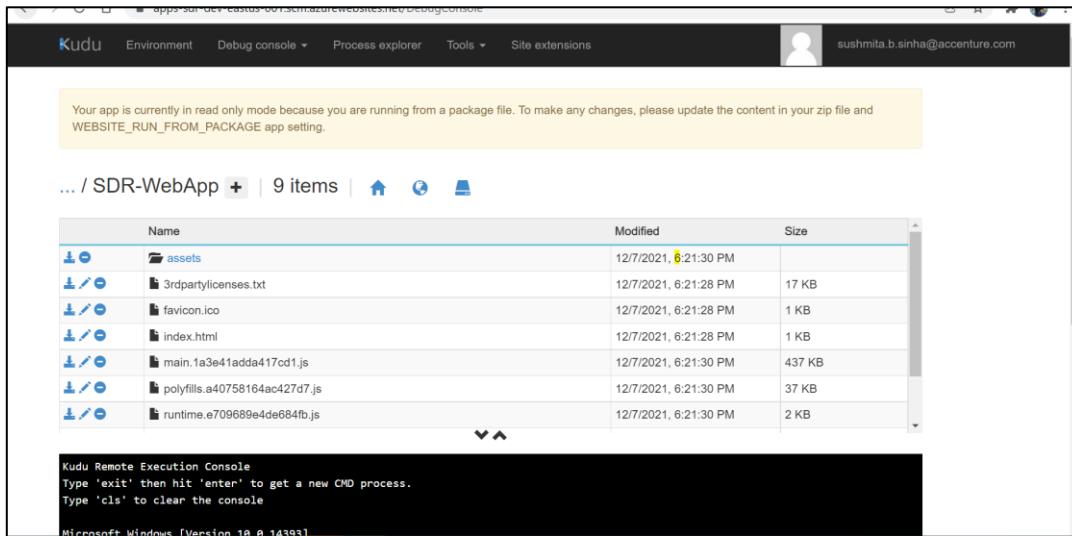
v. Go to Debug console and select CMD/Power Shell

Figure 58 Azure Kudu Tool



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

Figure 59 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:

- Secure APIs using client certificate authentication in API Management
- API Management uses client certificates to secure API access (i.e., client to API Management). It will validate certificates presented by the connecting client and compare certificate properties to desired values using policy expressions.

PRE-REQUISITES:

- Contributor access at Resource Group level.

CREATE CLIENT CERTIFICATE:

- Create self-signed certificate for authentication.

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

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

Figure 60 Client Certificate

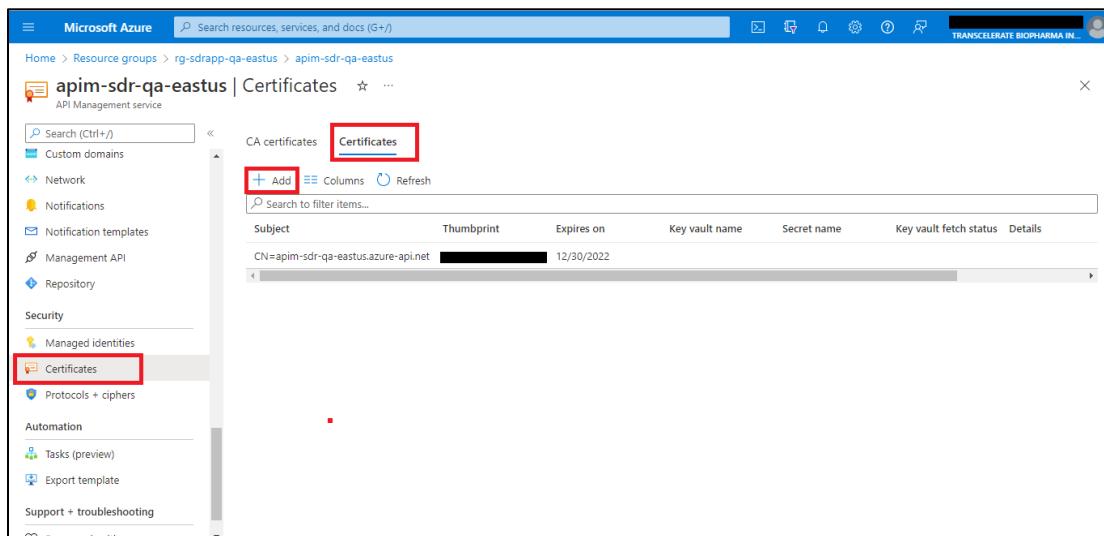


- iii. Export the certificate in .pfx format and set password when prompted.

UPLOAD THE CLIENT CERTIFICATE TO APIM:

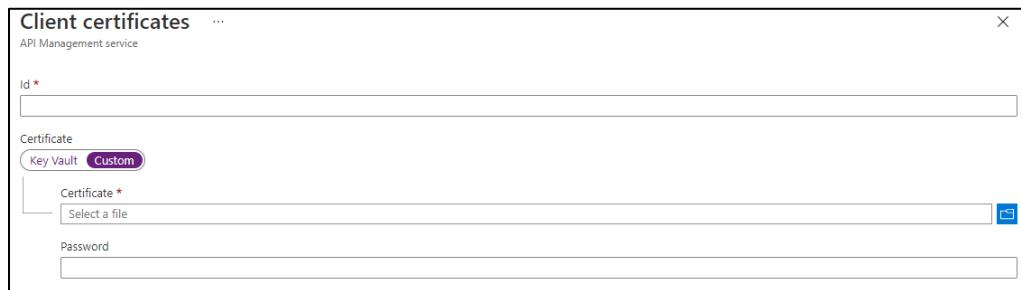
- i. In Azure Portal, Go to the Certificates option under the “Security” section of APIM. Go to “Certificates” option and click on “Add” option

Figure 61 APIM Certificates



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

Figure 62 APIM Certificates - Client Certificates

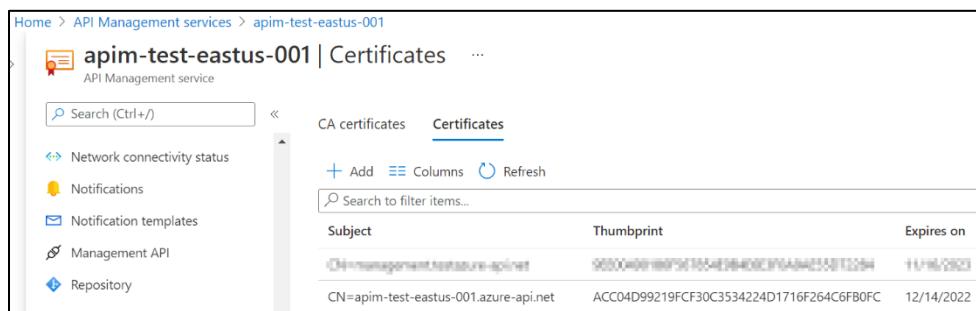


The screenshot shows a form titled 'Client certificates' under an 'API Management service'. It has fields for 'Id *' (a text input field), 'Certificate' (a dropdown menu with 'Key Vault' and 'Custom' options, currently set to 'Custom'), 'Certificate *' (a file upload input field with a placeholder 'Select a file'), and 'Password' (a text input field).

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

Client Certificate

Figure 63 APIM Certificates - Client Certificate

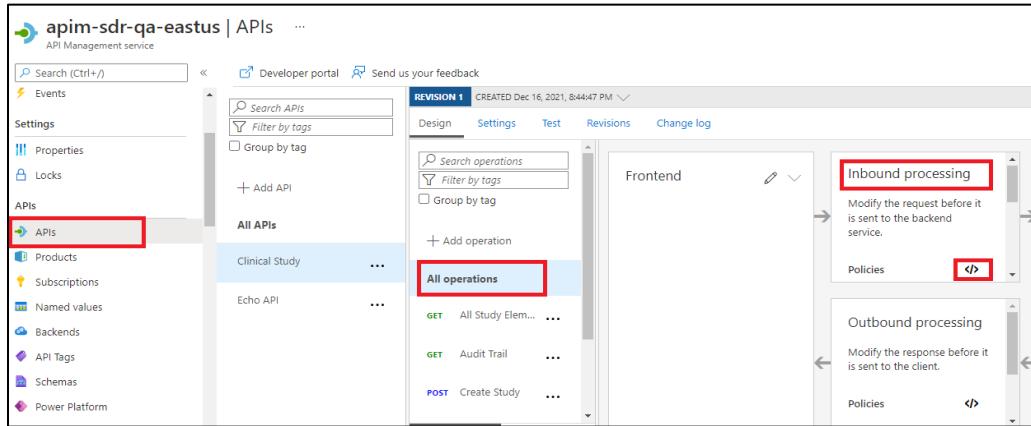


The screenshot shows the 'Certificates' blade for the 'apim-test-eastus-001' API Management service. On the left is a navigation sidebar with links like 'Network connectivity status', 'Notifications', 'Notification templates', 'Management API', and 'Repository'. The main area has tabs for 'CA certificates' and 'Certificates', with 'Certificates' selected. It includes buttons for '+ Add', 'Columns', and 'Refresh'. A search bar says 'Search to filter items...'. A table lists one certificate entry:

Subject	Thumbprint	Expires on
CN=management-testature-apinet	9E5004691B87967D54E0B400C910A0A232871294	11/16/2023
CN=apim-test-eastus-001.azure-api.net	ACC04D99219FCF30C3534224D1716F264C6FB0FC	12/14/2022

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

Figure 64 APIM Inbound Processing



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

```
<policies>
<inbound>
    <set-variable name="EmailAddress" value="@{
        string name = "EmptyAuthToken";
        var authHeader =
context.Request.Headers.GetValueOrDefault("Authorization",
"EmptyAuthToken");
        return authHeader.AsJwt()?.Claims.GetValueOrDefault("email",
"EmptyAuthToken");
    }" />
    <set-variable name="UserName" value="@{
        string name = "EmptyAuthToken";
        var authHeader =
context.Request.Headers.GetValueOrDefault("Authorization",
"EmptyAuthToken");
        return authHeader.AsJwt()?.Claims.GetValueOrDefault("name",
"EmptyAuthToken");
    }" />
    <choose>
        <when condition="@((context.Variables["EmailAddress"]) != null)">
            <trace source="My Global APIM Policy"
severity="information">
                <message>@(String.Format("{0} | {1}",
context.Api.Name, context.Operation.Name))</message>
                <metadata name="EmailAddress"
value="@((string)context.Variables["EmailAddress"])" />
                <metadata name="UserName"
value="@((string)context.Variables["UserName"])" />
            </trace>
        </when>
    </choose>
</inbound>
<outbound>
</outbound>
</policies>
```

```

<otherwise>
    <trace source="My Global APIM Policy"
severity="information">
        <message>@(String.Format("{0} | {1}",
context.Api.Name, context.Operation.Name))</message>
        <metadata name="EmailAddress" value="Not Available" />
        <metadata name="UserName" value="Not Available" />
    </trace>
</otherwise>
</choose>
<base />
<choose>
    <when condition="@(context.Request.Certificate == null ||
!context.Deployment.Certificates.Any(c => c.Value.Thumbprint ==
context.Request.Certificate.Thumbprint)
|| context.Request.Certificate.NotAfter<DateTime.Now)">
        <return-response>
            <set-status code="403" reason="Invalid client
certificate" />
        </return-response>
    </when>
</choose>
<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>

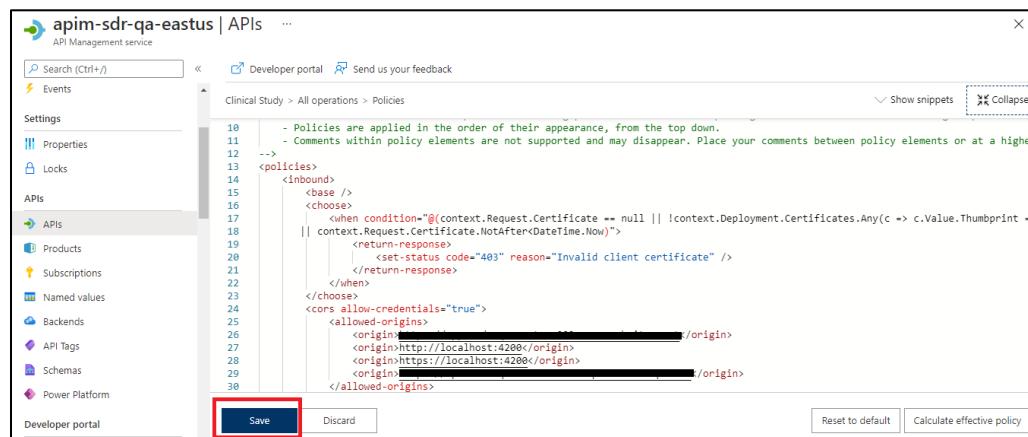
```

```

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

```

Figure 65 APIM - Inbound Policy



The screenshot shows the Azure API Management Policies editor for an API named "apim-sdr-qa-eastus". The left sidebar shows "APIs" selected. The main area displays the following XML code:

```

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 == context.Request.Certificate.NotAfter < DateTime.Now))">
18                  <return-response>
19                      <set-status code="403" reason="Invalid client certificate" />
20                  </return-response>
21              </when>
22          </choose>
23          <cors allow-credentials="true">
24              <allowed-origins>
25                  <origin>[REDACTED]/origin</origin>
26                  <origin>http://localhost:4200</origin>
27                  <origin>https://localhost:4200</origin>
28                  <origin>[REDACTED]</origin>
29              </allowed-origins>
30

```

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