Study Definition Repository (SDR) Reference Implementation

Infra Migration Guide (Release 0.5 to Release 3.0) Version 1.0

Document History

Version No.	Date	Author	Revision Description
V1.0	22-Apr-2024	ACN	• Added steps to create default group in Groups collection of Cosmos DB. • APIM stv2 platform migration steps included.

Table of Contents

List of figures

1 Introduction

1.1 Overview

This document details the steps required to migrate SDR Reference Implementation infrastructure from release Version 0.5 to Version 2.0.1 for users who have set up their own SDR instance for the Study Definition Repository – Reference Implementation on Azure Cloud Platform[1]. It provides details for deploying the new resources using azure portal. Additionally, it provides details of containerized deployment for the SDR RI API and UI applications.

1.2 Scope of Document

Scope of the document includes steps how to migrate SDR RI infrastructure from release 0.5 to release 3.0 on Azure Cloud.

1.3 Intended Audience

This document assumes a good understanding of Azure concepts and services. The audience for this document is users who have set up their own SDR instance (on Azure Cloud) and running version of SDR Release V0.5.

1.4 Definitions and Acronyms

Term / Abbreviation	Definition
API	Application Programming Interface
DB	Database
DDF	Digital Data Flow
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LLD	Low Level Design
REST	Representational State Transfer
SDR	Study Definition Repository
UI	User Interface
URL	Uniform Resource Locator
VNet	Virtual Network
CDISC USDM	Unified Study Definitions Model (USDM versioning is supporting major versions of CDISC USDM versions)
USDM Versioning	Hosting studies conformant to multiple versions of CDISC USDM in a single instance of SDR

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

Users are free to download the source code for the SDR from GitHub and design their own implementations. Those implementations can be on an environment of the user's choice, and do not have to be on Azure.

Change Audit	Capturing the audit information including user, date, time and action of study data CRUD activities

1.5 Out of Scope

This document does not include any instructions to upgrade SDR on Cloud platforms other than Azure. Nonetheless, the SDR Platform Agnostic Recommendations can be referred to for corresponding resource configurations to Infrastructure changes for Release V2.0.1.

2 Infrastructure changes for Release V3.0

2.1 Summary

As part of SDR Azure infrastructure migration, the below changes have been implemented and configured to upgrade from Release V0.5 to V3.0

- For Change Audit feature, below list of new resources have been added. Additionally, a new collection in CosmosDB has been created to hold change audit information.
    - Azure Service Bus
    - App Service Plan for Function App
    - Function App
    - Storage Account
- For USDM Versioning below changes have been done to APIM and Cosmos DB
    - New APIs have been implemented and APIM routes configured.
    - New common collection has been created to hold studies conformant to all USDM versions supported by SDR V3.0.
    - Additionally, Data updates have been done on USDM version 1.0 studies which will be covered in the data migration section.
- The certificate authentication has been removed for API endpoints supporting SDR UI and now is enabled only for SDR API endpoints.
- These are the resources that have been created to support Containerized Deployment. Existing App Service plans and App Services have been deleted.
    - Azure Container Registry
    - App Service Plan and App Service for UI App Service
    - App Service Plan and App Service for API App Service
- GitHub actions have been introduced to support containerized deployment. Containerized deployment is optional, and users can also use the existing standard way of deployment of build to the existing App services.
- Migration of APIM platform version from stv1 to stv2.
- Enabled connectivity between APIM and API App service through private endpoint.

The users who want to install a new version of SDR from scratch can export data or redo their SDR setup from scratch, they can refer to the platform setup and deployment guide.

Before that, users should take a backup of existing data. For data migration of existing data into new SDR setup user can use one of the recommended tools to export data collections and then import back to new environment.

2.2 References

Please refer to the documents below for information on latest SDR Architecture and Azure Resource Configurations.

Document Name	Document Link
SDR Solution Architecture	ddf-sdr-ri-solution-architecture-v6.0
Low Level Design	ddf-sdr-ri-platform-low-level-design-v6.0
Platform agnostic recommendations	ddf-sdr-ri-platform-agnostic-recommendations-v4.0
SDR Platform Setup and Deployment Guide	ddf-sdr-ri-platform-setup-and-deployment-guide-v7.0

3 Steps to Migrate

PRE-REQUISITES:

- SDR Running instance on Azure (SDR Release V0.5).
- Minimum Contributor level of access at Subscription Level.
- Optionally, basic understanding of containerized deployments.

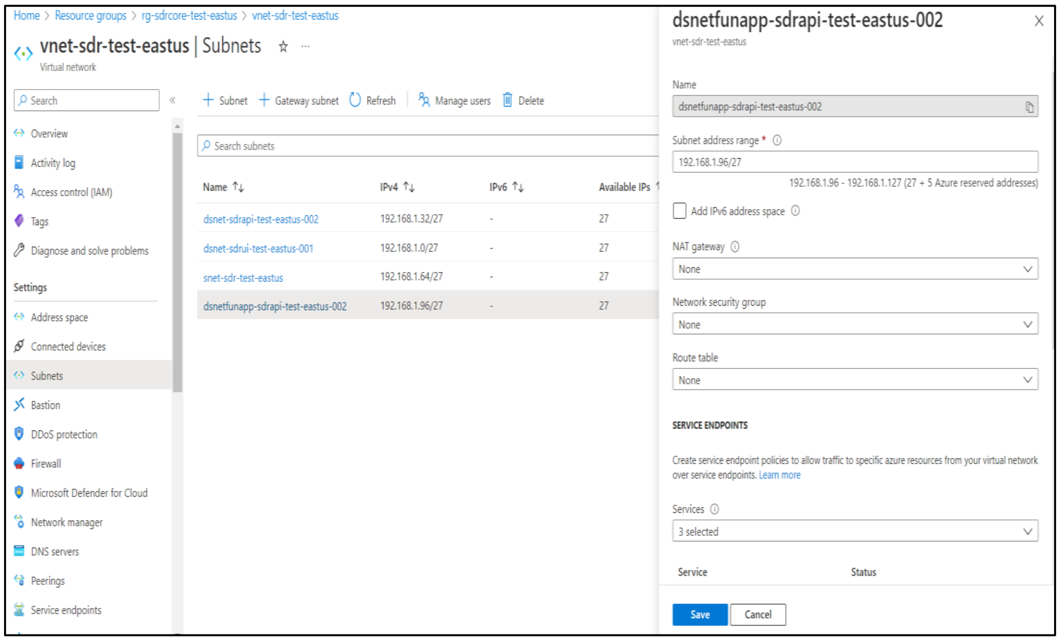
3.1 Resources Configurations for Change Audit Feature

3.1.1 Create Delegated Subnet

STEPS

1. Login to Azure Portal
2. Click on the resource groups tab.
3. Select the ***sdrcore*** resource group.
4. Refer the LLD and create a new delegated subnet.
   - Name
   - Subnet address range
   - Add IPv6 address space.
   - NAT gateway
   - Network Security Group
   - Route table
   - Services
   - Delegate subnet to a service

Figure 1 Delegated Subnet



vnet-sdr-test-eastus | Subnets

Virtual network

Name	IPv4	IPv6	Available IPs
dsnet-sdrapi-test-eastus-002	192.168.1.32/27	-	27
dsnet-sdrui-test-eastus-001	192.168.1.0/27	-	27
snet-sdr-test-eastus	192.168.1.64/27	-	27
dsnetfunapp-sdrapi-test-eastus-002	192.168.1.96/27	-	27

dsnetfunapp-sdrapi-test-eastus-002

vnet-sdr-test-eastus

Name: dsnetfunapp-sdrapi-test-eastus-002

Subnet address range *: 192.168.1.96/27

192.168.1.96 - 192.168.1.127 (27 + 5 Azure reserved addresses)

☐ Add IPv6 address space

NAT gateway: None

Network security group: None

Route table: None

SERVICE ENDPOINTS

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

Services: 3 selected

Service	Status

3.1.2 Create Azure Service Bus

Steps

1. Login to Azure Portal
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Refer the LLD document and create a new azure service bus.

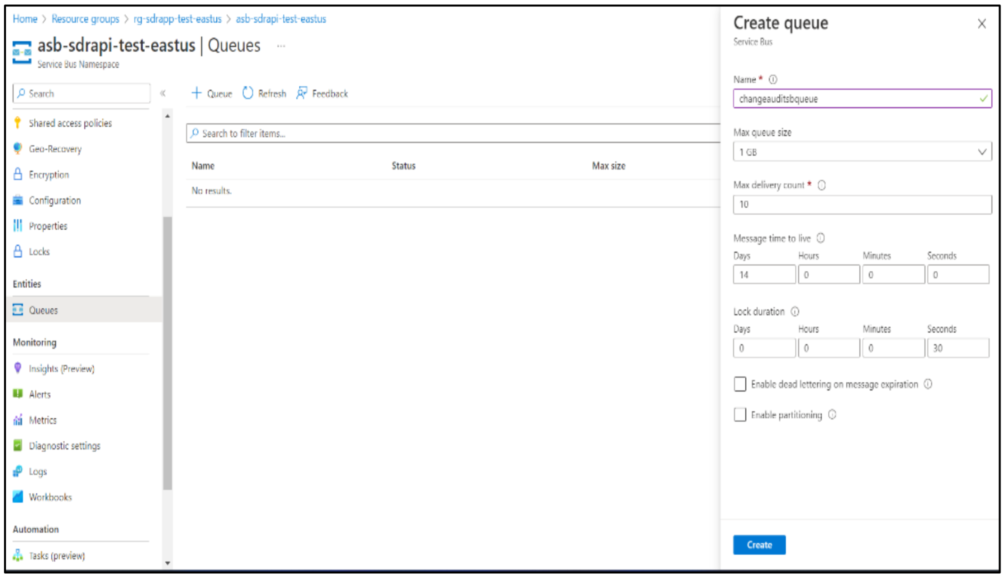
Service Bus Configuration

Steps

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Select the Service Bus Namespace and navigate to queues tab.
5. Create a Queue with name **(changeauditqueue)** then click on create.
6. Navigate to **changeauditqueue**, click on Shared Access Policies.
7. From Shared Access Policies add SAS Policy with Manage entity and capture the connection string values.
8. Go to Monitoring Tab and select Diagnostic Setting.
9. Click on add Diagnostic Settings and give the Diagnostic Setting Name.
10. Select Logs and Metrics and check “Send to Log Analytics Workspace”.
11. Save the settings.

Note: Please make a note of Queue name and connection string value required to be added in **Key Vaults Configurations Section 3.3.1 and Function App Application Settings Section 3.1.4.**

Figure 2 Service Bus Namespace Queue Configurations



Create queue
Service Bus

Name *: changeauditsbqueue

Max queue size: 1 GB

Max delivery count *: 10

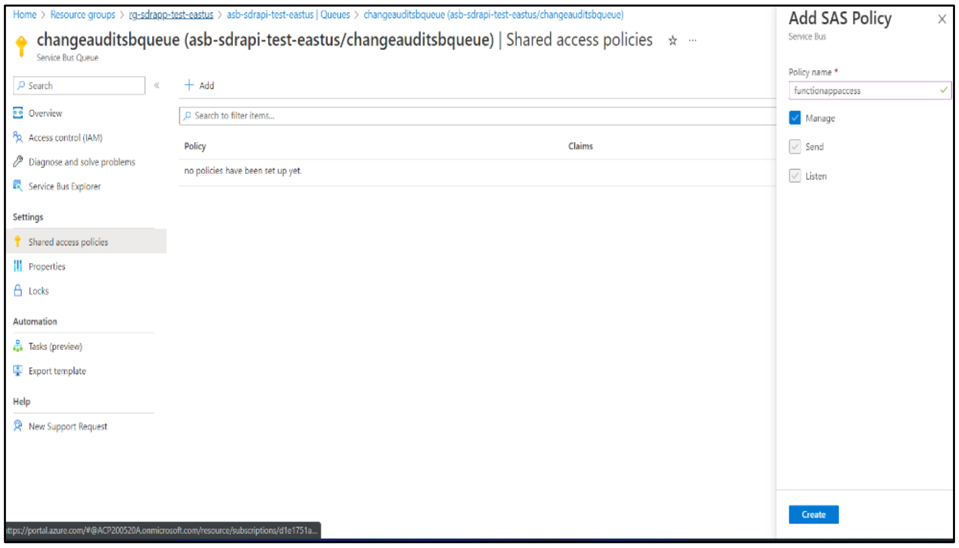
Message time to live: Days 14, Hours 0, Minutes 0, Seconds 0

Lock duration: Days 0, Hours 0, Minutes 0, Seconds 30

☐ Enable dead lettering on message expiration

☐ Enable partitioning

Figure 3 Service Bus Namespace Queue Shared access policies Configurations


Home > Resource groups > rg-sdrapp-test-eastus > asb-sdrapi-test-eastus | Queues > changeauditsbqueue (asb-sdrapi-test-eastus/changeauditsbqueue)
changeauditsbqueue (asb-sdrapi-test-eastus/changeauditsbqueue) | Shared access policies
Service Bus Queue
Search
Overview
Access control (IAM)
Diagnose and solve problems
Service Bus Explorer
Settings
Shared access policies
Properties
Locks
Automation
Tasks (preview)
Export template
Help
New Support Request
+ Add
Search to filter items...
Policy
Claims
no policies have been set up yet.
Create
ttps://portal.azure.com/#@ACP200520A.onmicrosoft.com/resource/subscriptions/d1e1751a...

Add SAS Policy
Service Bus

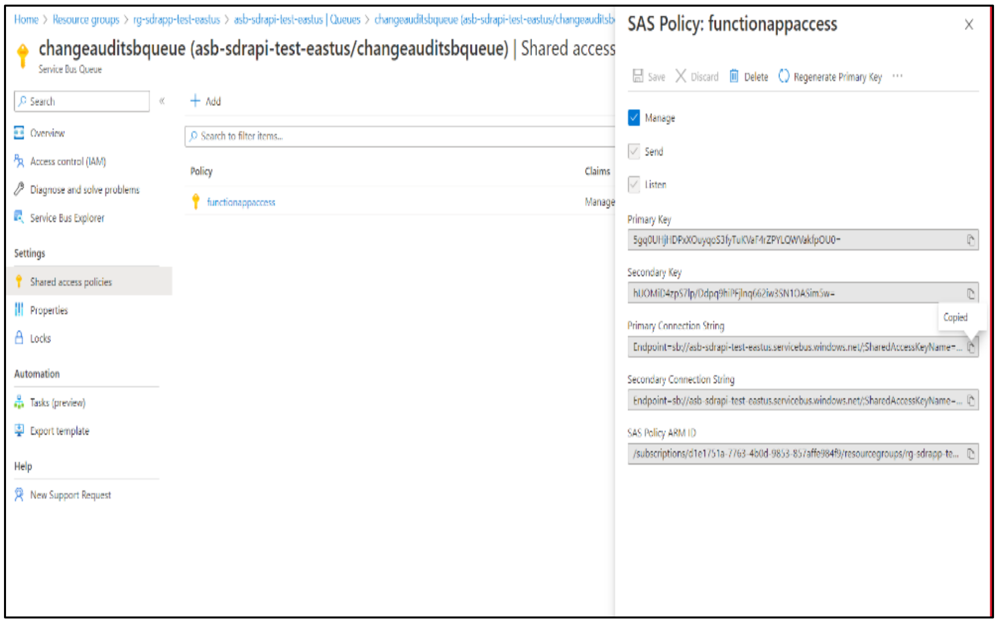
Policy name *: functionappaccess

☑ Manage

☑ Send

☑ Listen

Figure 4 Service Bus Namespace Queue Shared access policies


Home > Resource groups > rg-sdrapp-test-eastus > asb-sdrapi-test-eastus | Queues > changeauditsbqueue (asb-sdrapi-test-eastus/changeauditsb
changeauditsbqueue (asb-sdrapi-test-eastus/changeauditsbqueue) | Shared access
Service Bus Queue
Search
Overview
Access control (IAM)
Diagnose and solve problems
Service Bus Explorer
Settings
Shared access policies
Properties
Locks
Automation
Tasks (preview)
Export template
Help
New Support Request
Add
Search to filter items...
Policy
Claims
SAS Policy: functionappaccess
Save
Discard
Delete
Regenerate Primary Key
Copied

Policy	Claims
functionappaccess	Manage

[✓ Manage] [✓ Send] [✓ Listen]

Primary Key: 5gq0UHjHDPxXOuyqoS3fyTuKVaF4rZPYLOWVakfpOU0=

Secondary Key: hUOMiD4zpS7lp/Ddpq9hiPFjlnq662iw3SN1OASim5w=

Primary Connection String: Endpoint=sb://asb-sdrapi-test-eastus.servicebus.windows.net/;SharedAccessKeyName=...

Secondary Connection String: Endpoint=sb://asb-sdrapi-test-eastus.servicebus.windows.net/;SharedAccessKeyName=...

SAS Policy ARM ID: /subscriptions/d1e1751a-7763-4b0d-9853-857affe984f9/resourcegroups/rg-sdrapp-te...

3.1.3 Create App Service Plan for Function App

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Refer the LLD document and create a new app service plan for function app.

3.1.4 Create Function App

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Refer the LLD document and create a function app.

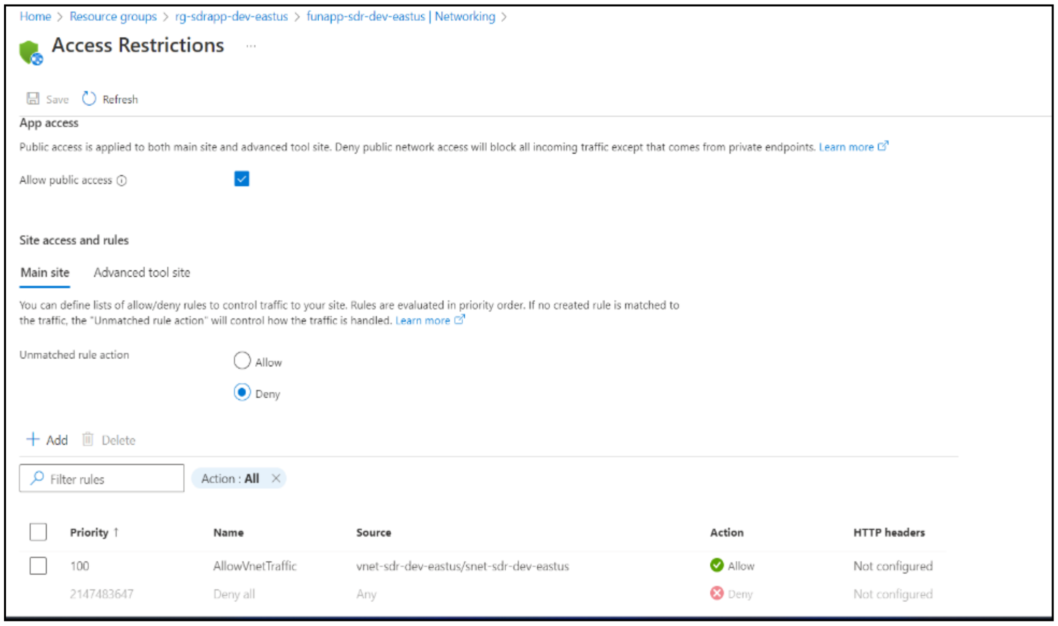
FUNCTION APP CONFIGURATION

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Select the Function App and navigate to networking tab.
5. Click on Access Restrictions and Add Rule
6. Click on VNet Integration and Add VNet and Subnet then click on Ok.
7. Go to Monitoring Tab and select Diagnostic Setting
8. Click on add Diagnostic Settings and give the Diagnostic Setting Name
9. Select Logs and Metrics and send to Log Analytics Workspace

10. Save the settings.
11. Go to Identity tab and enable System assigned managed identity and save the settings.
12. Navigate to Settings tab and click on Configuration tab.
13. Click on new application setting and add new Application Settings by using below mentioned table.
14. Save the settings.

Figure 5 Function App Inbound Traffic Configurations


Home > Resource groups > rg-sdrapp-dev-eastus > funapp-sdr-dev-eastus | Networking >

Access Restrictions

Save Refresh

App access

Public access is applied to both main site and advanced tool site. Deny public network access will block all incoming traffic except that comes from private endpoints. Learn more

Allow public access ✓

Site access and rules

Main site | Advanced tool site

You can define lists of allow/deny rules to control traffic to your site. Rules are evaluated in priority order. If no created rule is matched to the traffic, the "Unmatched rule action" will control how the traffic is handled. Learn more

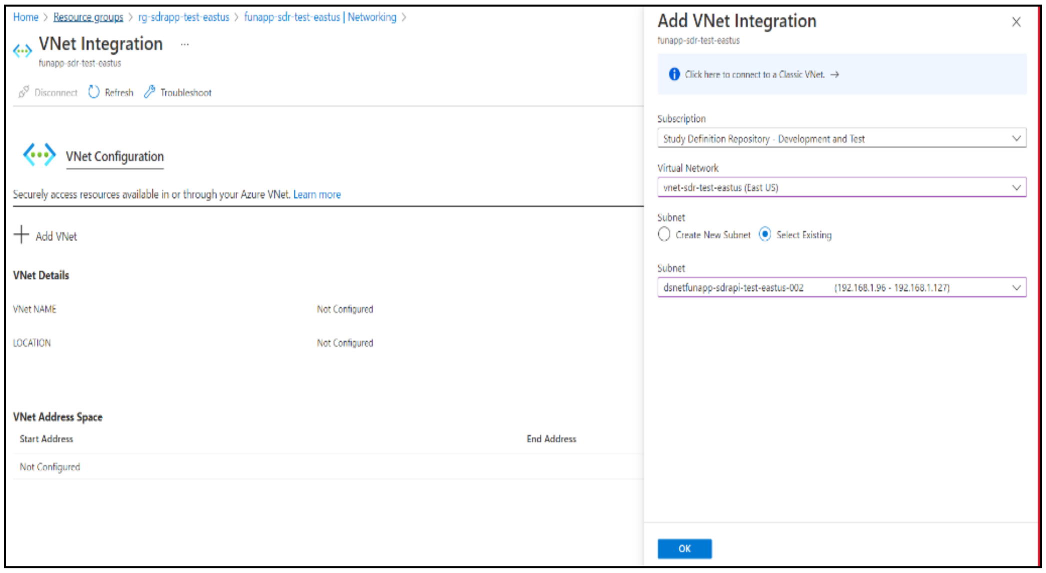
Unmatched rule action: Allow [✓ Deny]

Add Delete

Filter rules | Action : All

Priority ↑	Name	Source	Action	HTTP headers
100	AllowVnetTraffic	vnet-sdr-dev-eastus/snet-sdr-dev-eastus	✓ Allow	Not configured
2147483647	Deny all	Any	✕ Deny	Not configured

Figure 6 Function App Outbound Traffic Configurations


Home > Resource groups > rg-sdrapp-test-eastus > funapp-sdr-test-eastus | Networking >

VNet Integration
funapp-sdr-test-eastus

Disconnect Refresh Troubleshoot

VNet Configuration

Securely access resources available in or through your Azure VNet. Learn more

Add VNet

VNet Details

VNet NAME	Not Configured
LOCATION	Not Configured

VNet Address Space

Start Address	End Address
Not Configured	

Add VNet Integration
funapp-sdr-test-eastus

Click here to connect to a Classic VNet. →

Subscription: Study Definition Repository - Development and Test

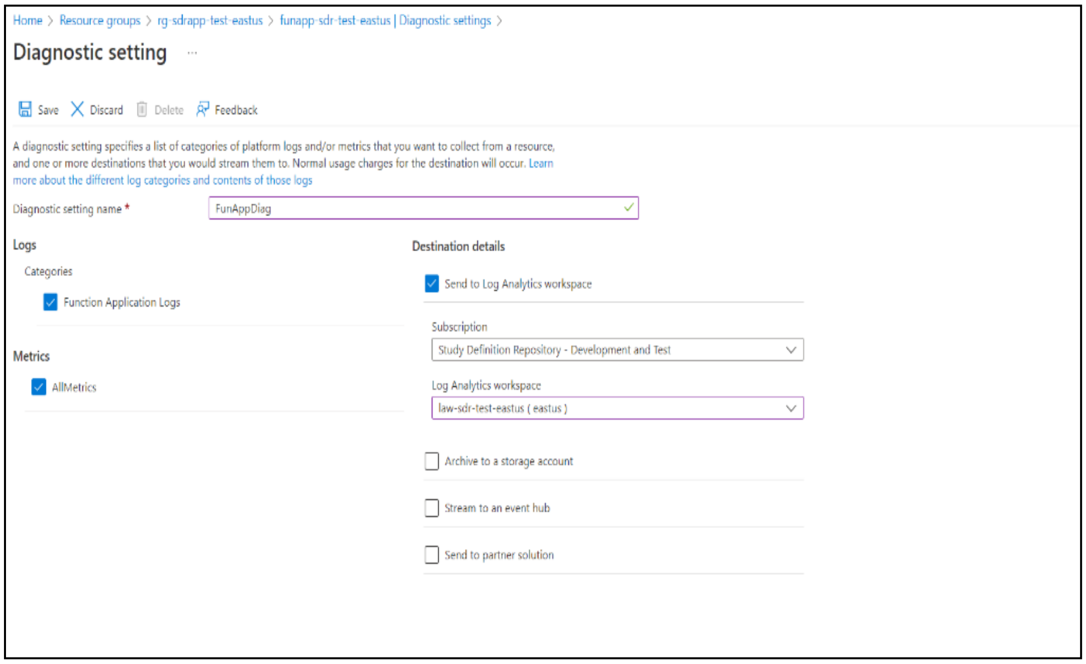
Virtual Network: vnet-sdr-test-eastus (East US)

Subnet: Create New Subnet [✓ Select Existing]

Subnet: dsnetfunapp-sdrapi-test-eastus-002 (192.168.1.96 - 192.168.1.127)

OK

Figure 7 Function App Diagnostic Configurations



Home > Resource groups > rg-sdrapp-test-eastus > funapp-sdr-test-eastus | Diagnostic settings >

Diagnostic setting

Save | Discard | Delete | Feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. Learn more about the different log categories and contents of those logs

Diagnostic setting name *: FunAppDiag

Logs

Categories

- [x] Function Application Logs

Metrics

- [x] AllMetrics

Destination details

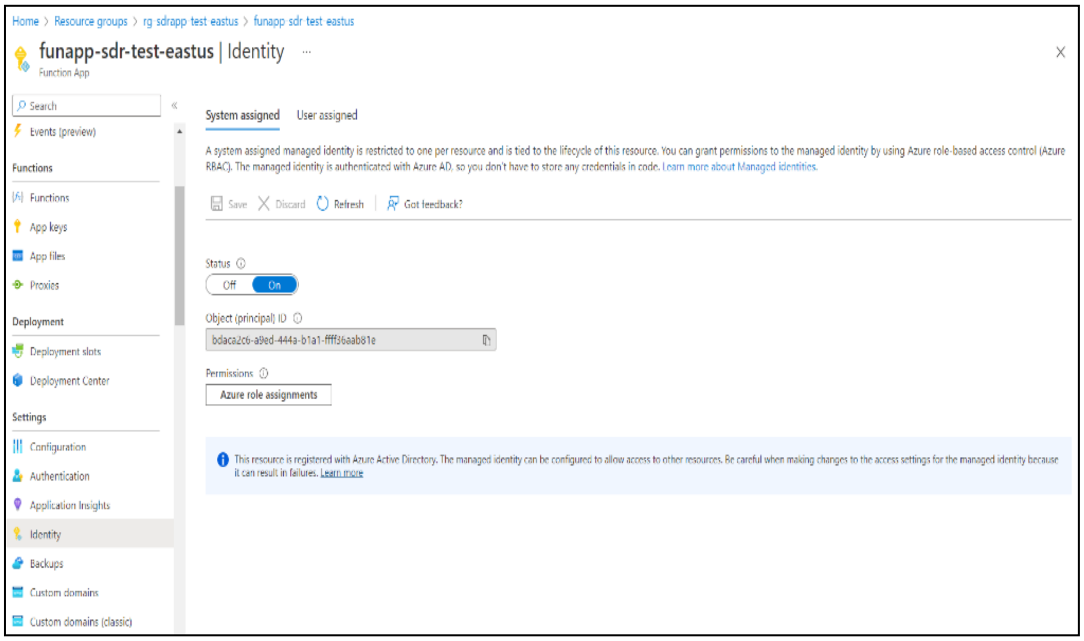
- [x] Send to Log Analytics workspace

Subscription: Study Definition Repository - Development and Test

Log Analytics workspace: law-sdr-test-eastus (eastus)

- [ ] Archive to a storage account
- [ ] Stream to an event hub
- [ ] Send to partner solution

Figure 8 Function App Identity Configurations



Home > Resource groups > rg sdrapp test eastus > funapp sdr test eastus

funapp-sdr-test-eastus | Identity

Function App

System assigned | User assigned

A system assigned managed identity is restricted to one per resource and is tied to the lifecycle of this resource. You can grant permissions to the managed identity by using Azure role-based access control (Azure RBAC). The managed identity is authenticated with Azure AD, so you don't have to store any credentials in code. Learn more about Managed identities.

Save | Discard | Refresh | Got feedback?

Status: Off [✓ On]

Object (principal) ID: bdaca2c6-a9ed-444a-b1a1-ffff36aab81e

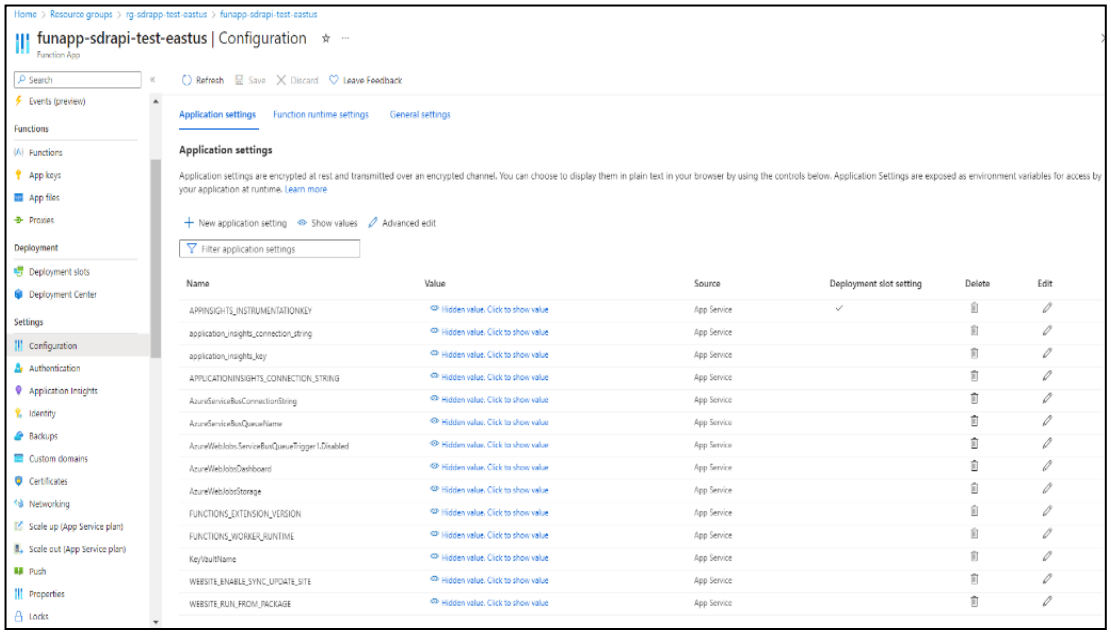
Permissions: Azure role assignments

This resource is registered with Azure Active Directory. The managed identity can be configured to allow access to other resources. Be careful when making changes to the access settings for the managed identity because it can result in failures. Learn more

Function App Application Settings

Name	Value
AzureServiceBusConnectionString	Refer to Service_Bus_Configuration Section
AzureServiceBusQueueName	Provide Azure Service Bus Queue Name
KeyVaultName	Provide Key Vault URL form Key Vault

Figure 9 Function App new Application settings



Application settings

Application settings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below. Application Settings are exposed as environment variables for access by your application at runtime. Learn more

Name	Value	Source	Deployment slot setting	Delete	Edit
APPINSIGHTS_INSTRUMENTATIONKEY	Hidden value. Click to show value	App Service	✓	🗑	✎
application_insights_connection_string	Hidden value. Click to show value	App Service		🗑	✎
application_insights_key	Hidden value. Click to show value	App Service		🗑	✎
APPLICATIONINSIGHTS_CONNECTION_STRING	Hidden value. Click to show value	App Service		🗑	✎
AzureServiceBusConnectionString	Hidden value. Click to show value	App Service		🗑	✎
AzureServiceBusQueueName	Hidden value. Click to show value	App Service		🗑	✎
AzureWebJobs.ServiceBusQueueTrigger1.Disabled	Hidden value. Click to show value	App Service		🗑	✎
AzureWebJobsDashboard	Hidden value. Click to show value	App Service		🗑	✎
AzureWebJobsStorage	Hidden value. Click to show value	App Service		🗑	✎
FUNCTIONS_EXTENSION_VERSION	Hidden value. Click to show value	App Service		🗑	✎
FUNCTIONS_WORKER_RUNTIME	Hidden value. Click to show value	App Service		🗑	✎
KeyVaultName	Hidden value. Click to show value	App Service		🗑	✎
WEBSITE_ENABLE_SYNC_UPDATE_SITE	Hidden value. Click to show value	App Service		🗑	✎
WEBSITE_RUN_FROM_PACKAGE	Hidden value. Click to show value	App Service		🗑	✎

3.1.5 Changes to Cosmos DB Collections

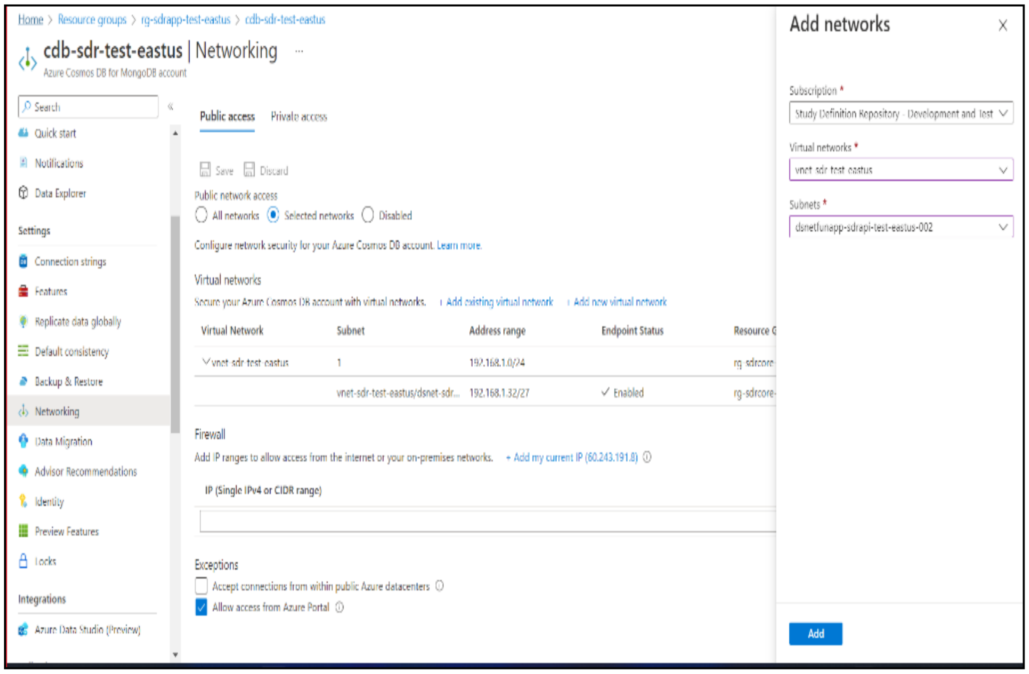
STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Select Azure Cosmos DB for MongoDB and navigate to networking tab.
5. Go to public access and select Selected networks, under existing VNet, add a new subnet with name as per the naming convention (E.g., **dsnetfunapp-sdrapi-qa-eastus-002)** then save the settings.
6. Go to Data Explorer and click on New Collection and give the existing database i.e., SDR and give the Collection id (**ChangeAudit)** then click on Ok.
7. Navigate to **ChangeAudit** data Collection and click on settings.
8. Click on add indexing Policy and add index as per below table.

Index Policy table

Definition	*Type*
_id	*Single field*

Figure 10 Network Configurations for Azure Cosmos DB for MongoDB


Home > Resource groups > rg-sdrapp-test-eastus > cdb-sdr-test-eastus

cdb-sdr-test-eastus | Networking
Azure Cosmos DB for MongoDB account

Public network access: All networks [✓ Selected networks] Disabled

Configure network security for your Azure Cosmos DB account. Learn more.

Virtual networks

Secure your Azure Cosmos DB account with virtual networks. + Add existing virtual network + Add new virtual network

Virtual Network	Subnet	Address range	Endpoint Status	Resource G
vnet-sdr-test-eastus	1	192.168.1.0/24		rg-sdrcore
	vnet-sdr-test-eastus/dsnet-sdr...	192.168.1.32/27	✓ Enabled	rg-sdrcore-

Firewall

Add IP ranges to allow access from the internet or your on-premises networks. + Add my current IP (60.243.191.8)

IP (Single IPv4 or CIDR range)

Exceptions

- [ ] Accept connections from within public Azure datacenters
- [x] Allow access from Azure Portal

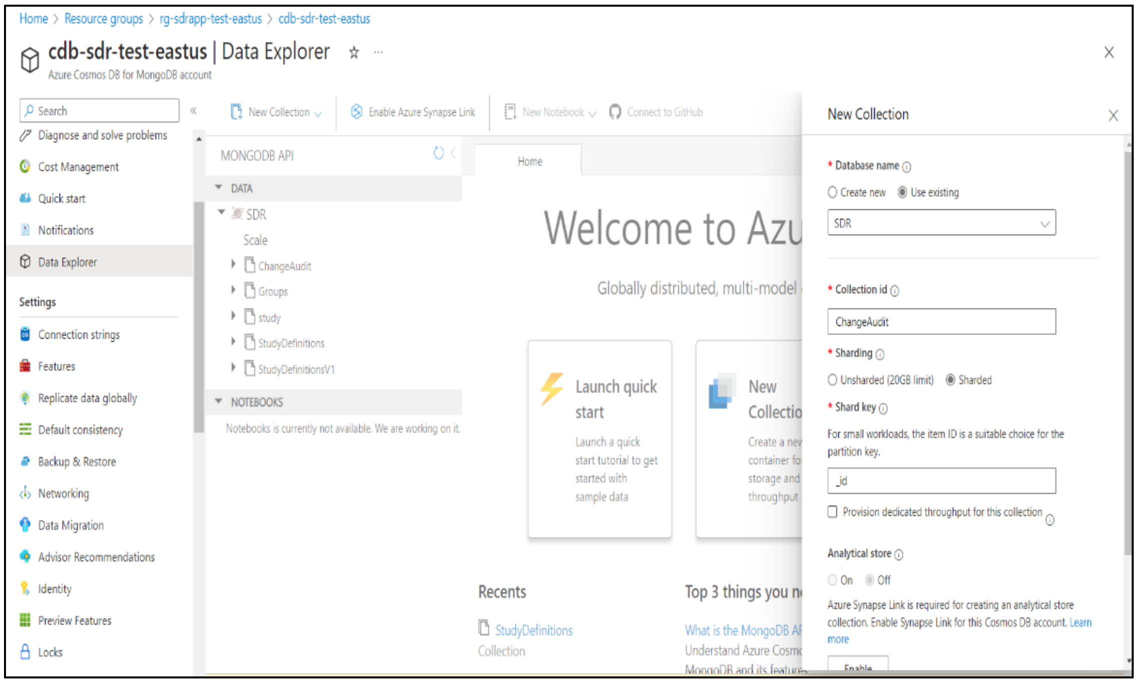
Add networks

Subscription *: Study Definition Repository - Development and Test

Virtual networks *: vnet-sdr-test-eastus

Subnets *: dsnetfunapp-sdrapi-test-eastus-002

Figure 11 Adding ChangeAudit Data Collections


Home > Resource groups > rg-sdrapp-test-eastus > cdb-sdr-test-eastus

cdb-sdr-test-eastus | Data Explorer
Azure Cosmos DB for MongoDB account

Notebooks is currently not available. We are working on it.

New Collection

* Database name: Create new [✓ Use existing]

SDR

* Collection id: ChangeAudit

* Sharding: Unsharded (20GB limit) [✓ Sharded]

* Shard key

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

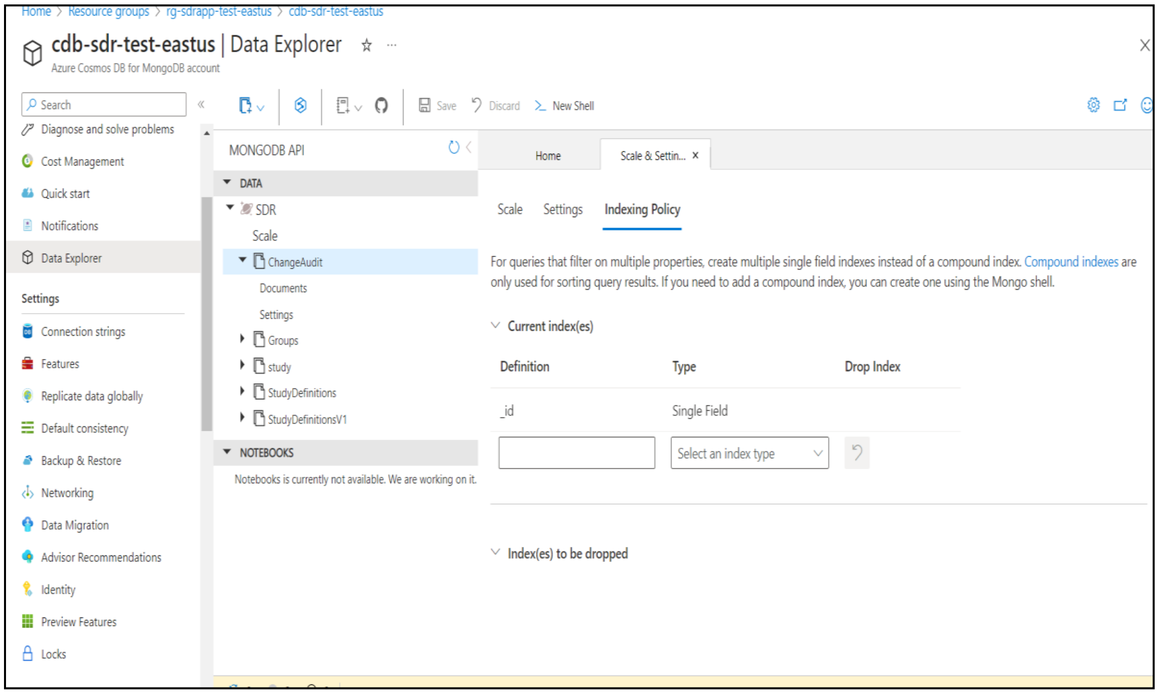
_id

- [ ] Provision dedicated throughput for this collection

Analytical store: On [✓ Off]

Azure Synapse Link is required for creating an analytical store collection. Enable Synapse Link for this Cosmos DB account. Learn more

Figure 12 Adding Indexing Policy for ChangeAudit Data Collection



Notebooks is currently not available. We are working on it.

For queries that filter on multiple properties, create multiple single field indexes instead of a compound index. Compound indexes are only used for sorting query results. If you need to add a compound index, you can create one using the Mongo shell.

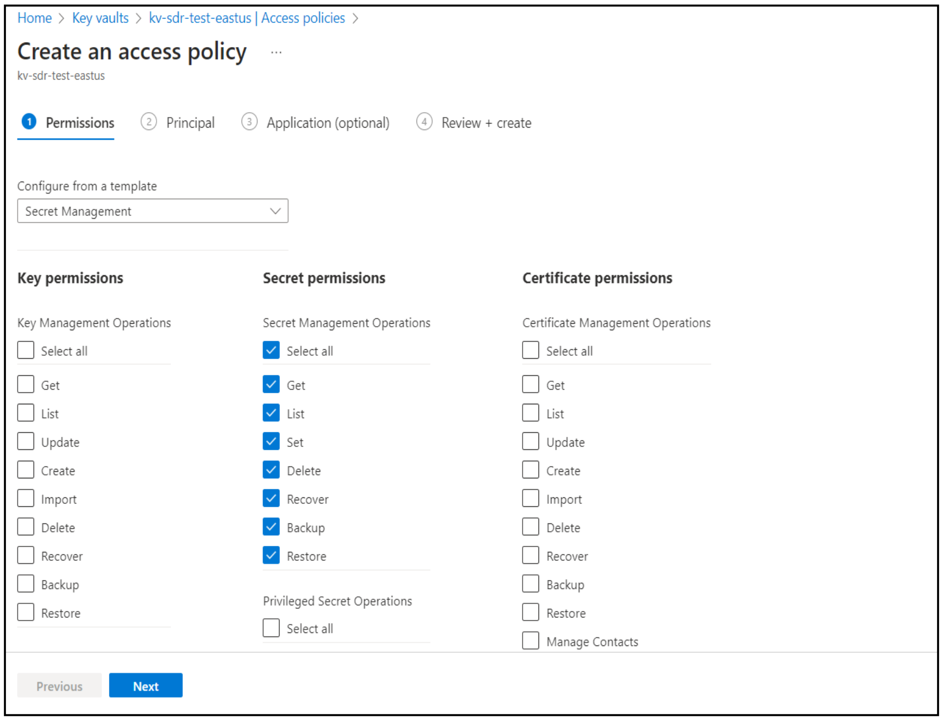
Definition	Type	Drop Index
_id	Single Field	
	Select an index type	

3.1.6 Key Vault access for function app

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrcore*** resource group.
4. Select Azure Key Vault and go to Access policies tab.
5. Click on Create and select Secret Management Configure from a template.
6. Click on Next and search for function app name and select Function App (E.g., funapp-sdrapi-qa-eastus).
7. Click on Next and click on Create.

Figure 13 Providing Access Policy for Function App



Home > Key vaults > kv-sdr-test-eastus | Access policies >

Create an access policy ...

kv-sdr-test-eastus

1 Permissions 2 Principal 3 Application (optional) 4 Review + create

Configure from a template: Secret Management

Key permissions

Key Management Operations

- [ ] Select all
- [ ] Get
- [ ] List
- [ ] Update
- [ ] Create
- [ ] Import
- [ ] Delete
- [ ] Recover
- [ ] Backup
- [ ] Restore

Secret permissions

Secret Management Operations

- [✓] Select all
- [✓] Get
- [✓] List
- [✓] Set
- [✓] Delete
- [✓] Recover
- [✓] Backup
- [✓] Restore

Privileged Secret Operations

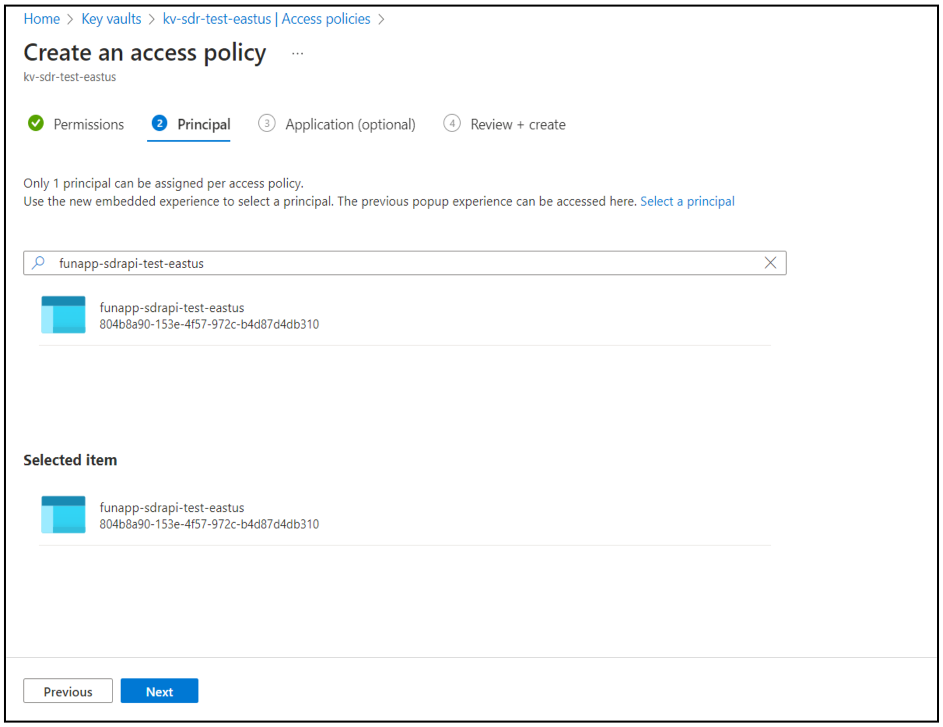
- [ ] Select all

Certificate permissions

Certificate Management Operations

- [ ] Select all
- [ ] Get
- [ ] List
- [ ] Update
- [ ] Create
- [ ] Import
- [ ] Delete
- [ ] Recover
- [ ] Backup
- [ ] Restore
- [ ] Manage Contacts

Previous Next



Home > Key vaults > kv-sdr-test-eastus | Access policies >

Create an access policy ...

kv-sdr-test-eastus

✓ Permissions 2 Principal 3 Application (optional) 4 Review + create

Only 1 principal can be assigned per access policy.
Use the new embedded experience to select a principal. The previous popup experience can be accessed here. Select a principal

funapp-sdrapi-test-eastus

funapp-sdrapi-test-eastus
804b8a90-153e-4f57-972c-b4d87d4db310

Selected item

funapp-sdrapi-test-eastus
804b8a90-153e-4f57-972c-b4d87d4db310

Previous Next

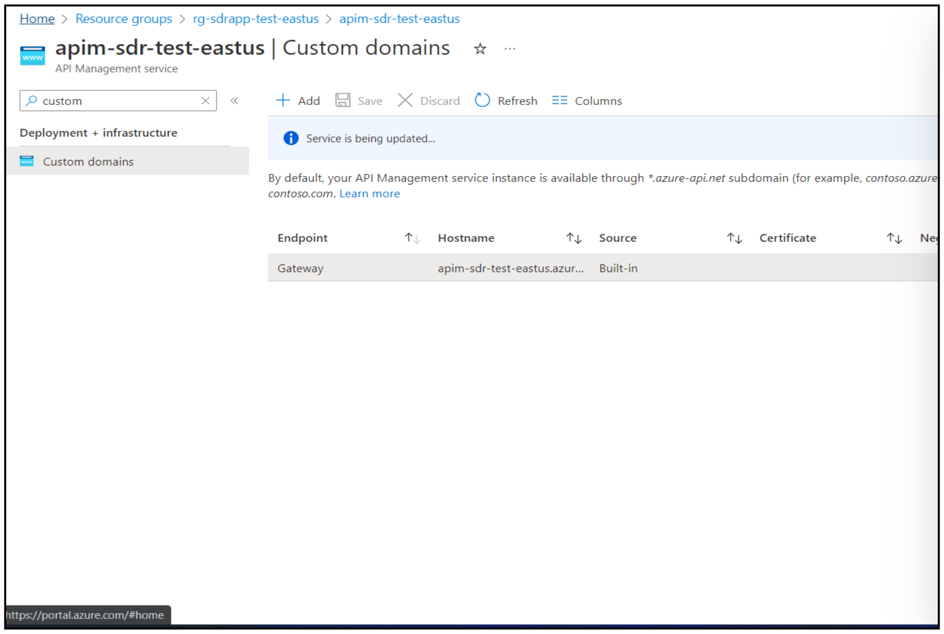
3.2 Resources Configuration for USDM Versioning

3.2.1 Disable Client Certificate in APIM Custom Domain

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Select the API Management service and search for Custom domains.
5. Go to Custom domains and unselect Negotiate client certificate.
6. Update the settings.

Figure 14 Custom domains Configuration


Home > Resource groups > rg-sdrapp-test-eastus > apim-sdr-test-eastus
apim-sdr-test-eastus | Custom domains
API Management service
custom
Deployment + infrastructure
Custom domains
Add
Save
Discard
Refresh
Columns

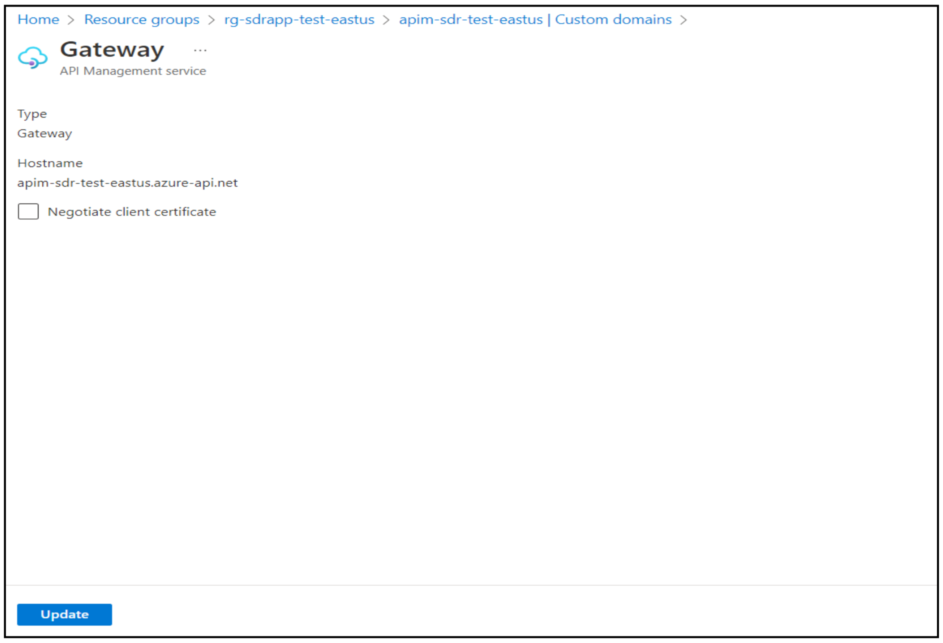
Service is being updated...

By default, your API Management service instance is available through *.azure-api.net subdomain (for example, contoso.azure contoso.com. Learn more

Endpoint	Hostname	Source	Certificate	Ne
Gateway	apim-sdr-test-eastus.azur...	Built-in		

https://portal.azure.com/#home

Figure 15 Custom domains Configuration – Uncheck Negotiate Client Certificate


Home > Resource groups > rg-sdrapp-test-eastus > apim-sdr-test-eastus | Custom domains >

Gateway ...
API Management service

Type
Gateway

Hostname
apim-sdr-test-eastus.azure-api.net

☐ Negotiate client certificate

Update

3.2.2 Migrate API Management service to stv2 platform

To migrate the existing API Management service from stv1 to stv2, the network configuration must be updated with subnet attached to a network security group and public IP.

- **Create Network Security Group**

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group section.
4. Refer the LLD document and create a new Network security group with mentioned inbound/outbound security rules.

- **Create a new Subnet**

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrcore*** resource group section.
4. Select the existing VNet resource, add a new subnet and attach the network security group created in previous step and save.

- **Create Public IP resource**

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group section.
4. Refer the LLD document and create a new Public IP resource.

- **Update VNet configuration of APIM service**

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group section.
4. Open API Management resource and navigate to Networking blade.
5. Update the virtual network configuration to point to newly created subnet and Public IP.
6. Click on Save. This action will trigger stv2 platform migration and a message will be show "Service is being updated". It will take approximately 30-45mins to complete.
7. Once the migration is completed, the message will disappear. Now the platform version will appear as stv2.x in the overview blade of API Management resource.

3.2.3 APIM Route Configuration

First, user must clear all existing routes and reconfigure the routes as per the table below. One example is given below to configure route and same steps should be followed for all the routes.

Method	API Name	Name	APIM Route	URL	Responses
GET	SDR API	Common - Get Study History	/api	/studydefinitions/history	200 OK
GET	SDR API	Common - Get Revision History	/api	/studydefinitions/{studyId}/revisionhistory	200 OK
GET	SDR API	Common - Get Change Audit	/api	/studydefinitions/{studyId}/changeaudit	200 OK
GET	SDR API	Common - Get API Versions	/api	/versions	200 OK
GET	SDR API	Common - Get Study Raw Data	/api	/studydefinitions/{studyId}/rawdata	200 OK
GET	SDR UI Admin	Check Group Name	/api/ui/admin	/usergroups/checkgroupname	200 OK

POST	SDR UI Admin	Get Group List	/api/ui/admin	/usergroups/getgroups	200 OK
GET	SDR UI Admin	Get Groups	/api/ui/admin	/usergroups/getgrouplist	200 OK
POST	SDR UI Admin	Get Users	/api/ui/admin	/usergroups/getusers	200 OK
GET	SDR UI Admin	List Users	/api/ui/admin	/usergroups/listusers	200 OK
POST	SDR UI Admin	Post Group	/api/ui/admin	/usergroups/postgroup	200 OK
POST	SDR UI Admin	Post User	/api/ui/admin	/usergroups/postuser	200 OK
POST	SDR UI	Usage Reports	/api/ui	/reports/usage	200 OK
POST	SDR UI	Search API	/api/ui	/studydefinitions/search	200 OK
POST	SDR UI	Search Study Title	/api/ui	/studydefinitions/searchstudytitle	200 OK
GET	SDR UI	Common - Get Version History	/api/ui	/studydefinitions/{studyId}/auditTrail	200 OK
GET	SDR UI	V1 Get Study Definition	/api/ui	/v1/studydefinitions/{studyId}	200 OK
GET	SDR UI	V2 Get Study Definition	/api/ui	/v2/studydefinitions/{studyId}	200 OK
GET	SDR UI	Get Study Links	/api/ui	/studydefinitions/{studyId}/links	200 OK
GET	SDR UI	V2 Get Study Design SOA	/api/ui	/v2/studydefinitions/{studyId}/studydesigns/soa	200 OK
GET	SDR API	V1 Get Study Definition	/api	/v1/studydefinitions/{studyId}	200 OK
POST	SDR API	V1 Post Study Definition	/api	/v1/studydefinitions	201 Created
GET	SDR API	V1 Get Study Design	/api	/v1/studydesigns	200 OK
GET	SDR API	V2 Get Study Definition	/api	/v2/studydefinitions/{studyId}	200 OK
POST	SDR API	V2 Put Study Definition	/api	/v2/studydefinitions	201 Created
PUT	SDR API	V2 Post Study Definition	/api	/v2/studydefinitions	201 Created
GET	SDR API	V2 Get Study Design	/api	/v2/studydesigns	200 OK

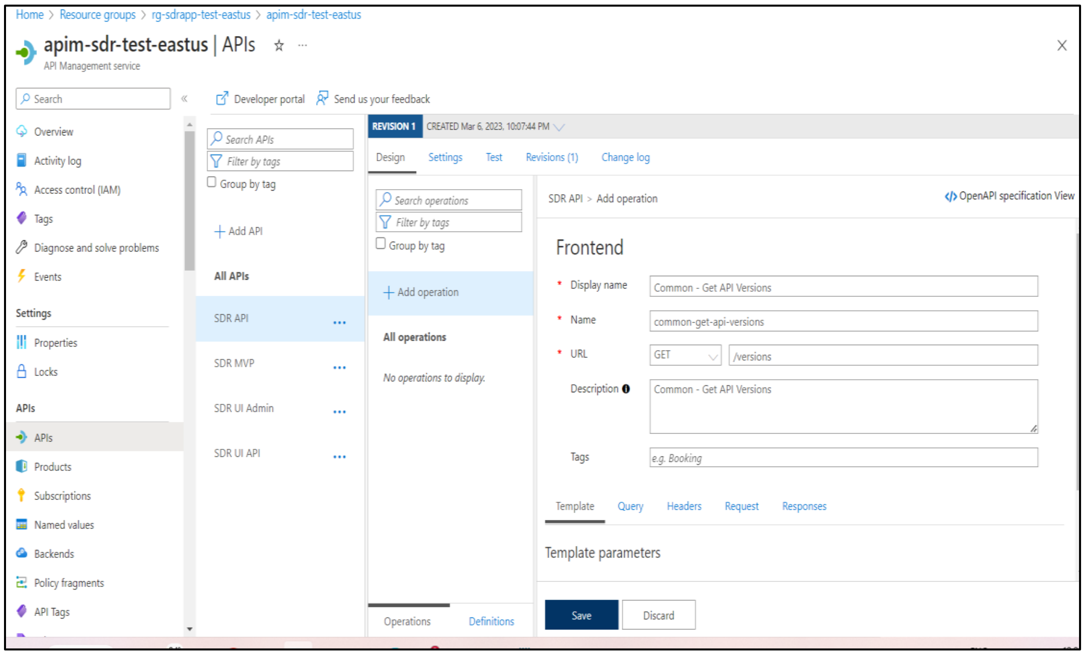
GET	SDR API	V2 Get Study Design SOA	/api	/v2/studydefinitions/{studyId}/studydesigns/soa	200 OK
GET	SDR API	V2 Get eCPT	/api	/v3/studyDefinitions/{studyId}/studydesigns/eCPT	200 OK
GET	SDR API	V3 Get Study Definition	/api	/v3/studydefinitions/{studyId}	200 OK
POST	SDR API	V3 Put Study Definition	/api	/v3/studydefinitions	201 Created
PUT	SDR API	V3 Post Study Definition	/api	/v3/studydefinitions	201 Created
GET	SDR API	V3 Get Study Design	/api	/v3/studydesigns	200 OK
GET	SDR API	V3 Get Study Design SOA	/api	/v3/studydefinitions/{studyId}/studydesigns/soa	200 OK
GET	SDR API	V3 Get eCPT	/api	/v3/studyDefinitions/{studyId}/studydesigns/eCPT	200 OK
POST	SDR API	V3 Validate USDM Conformance	/api	/v3/studyDefinitions/validate-usdm-conformance	200 OK
GET	SDR API	V3 Get Version Comparison	/api	/v3/studyDefinitions/{studyId}/version-comparison	200 OK

- **SDR API Route Configuration**

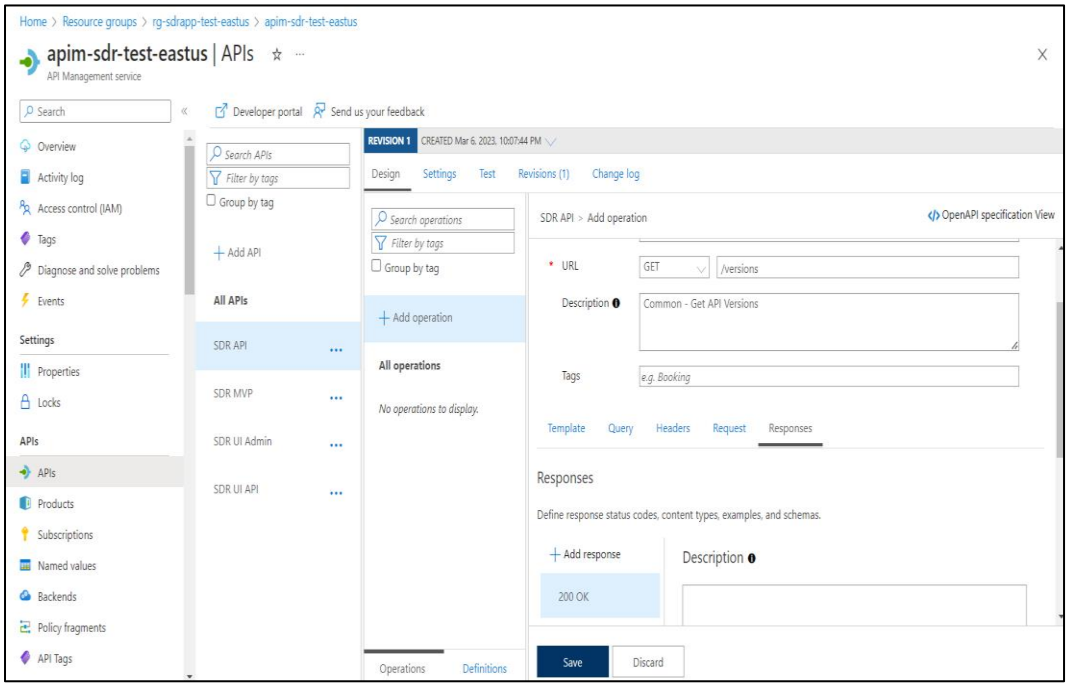
STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group ration.
4. Select the API Management service and navigate to APIs Tab.
5. Select SDR API and click on Add operation.
6. Provide the details as per table above.
7. Save the settings.
    i. Display Name
    ii. Name
    iii. URL
    iv. Responses
8. Follow the same steps for all the routes.

Figure 16 SDR API Route Configuration

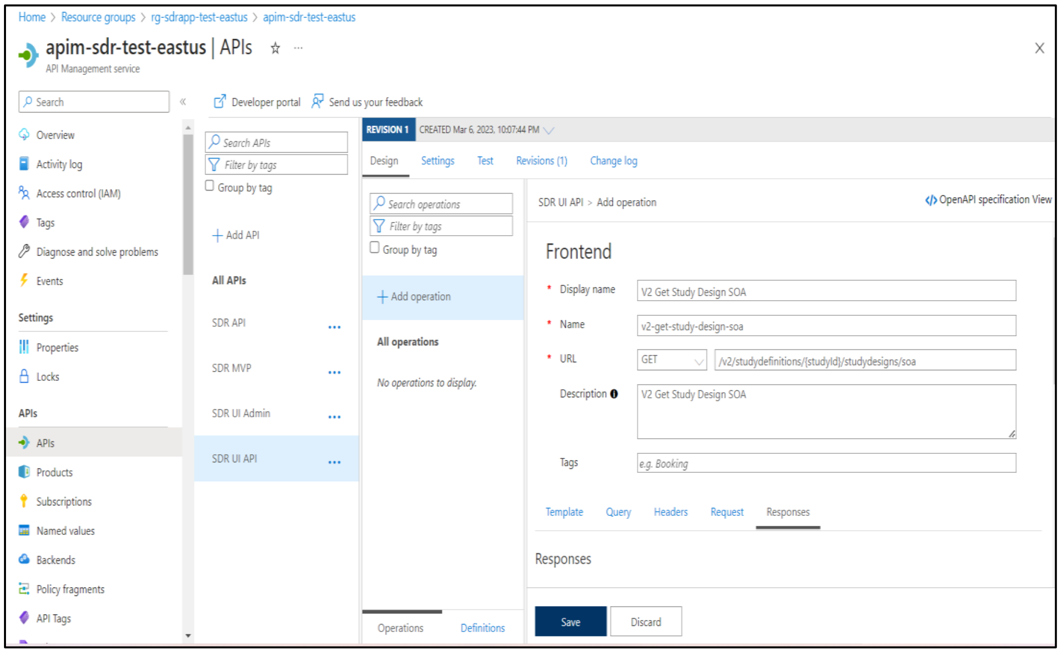


* Display name	Common - Get API Versions
* Name	common-get-api-versions
* URL	GET /versions
Description	Common - Get API Versions
Tags	e.g. Booking

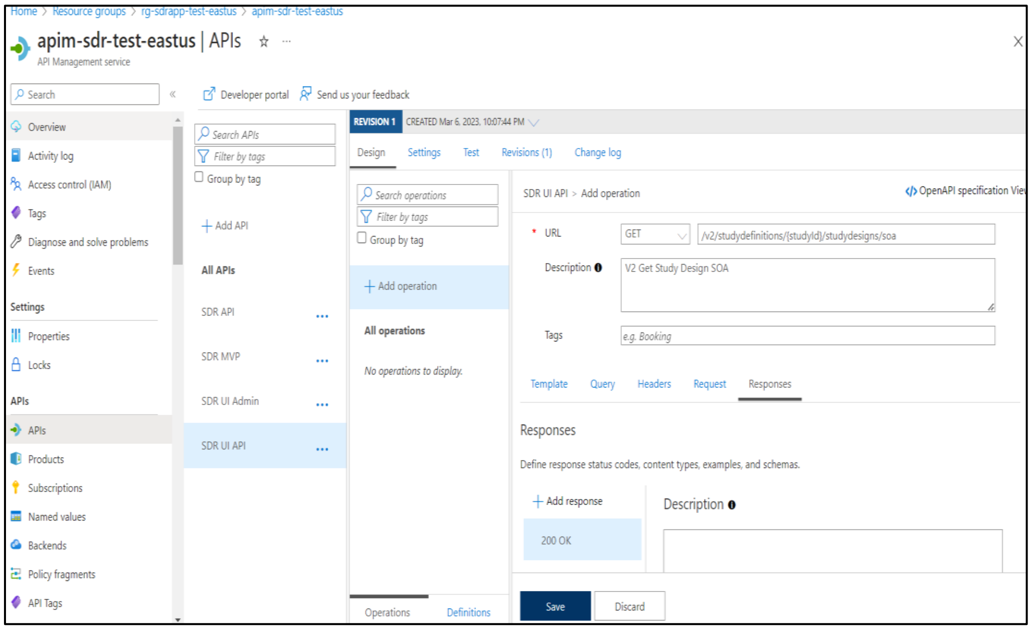


* URL	GET /versions
Description	Common - Get API Versions
Tags	e.g. Booking

Figure 17 SDR UI Route Configuration



Display name: V2 Get Study Design SOA
Name: v2-get-study-design-soa
URL: GET /v2/studydefinitions/{studyId}/studydesigns/soa
Description: V2 Get Study Design SOA
Tags: e.g. Booking



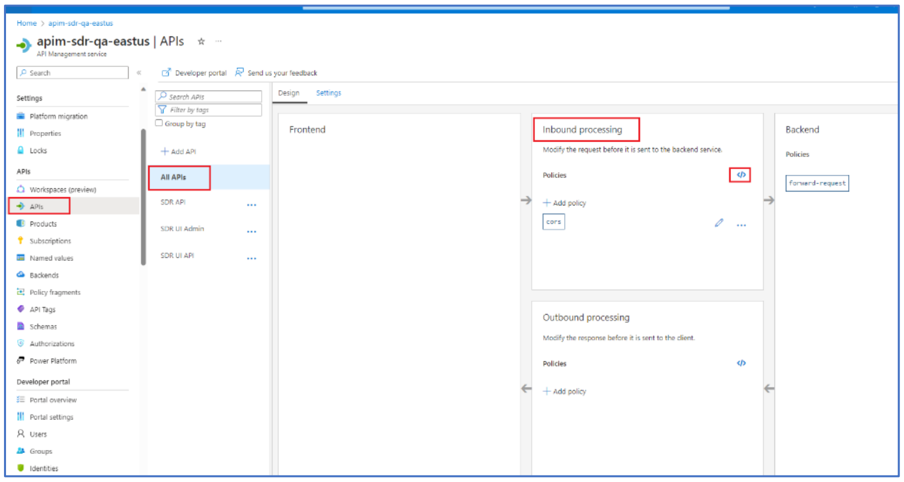
URL: GET /v2/studydefinitions/{studyId}/studydesigns/soa
Description: V2 Get Study Design SOA
Tags: e.g. Booking

Define response status codes, content types, examples, and schemas.

3.2.4 Configure Inbound Policies for All APIs

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

Figure 18 : APIM Inbound Policy for All APIs



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

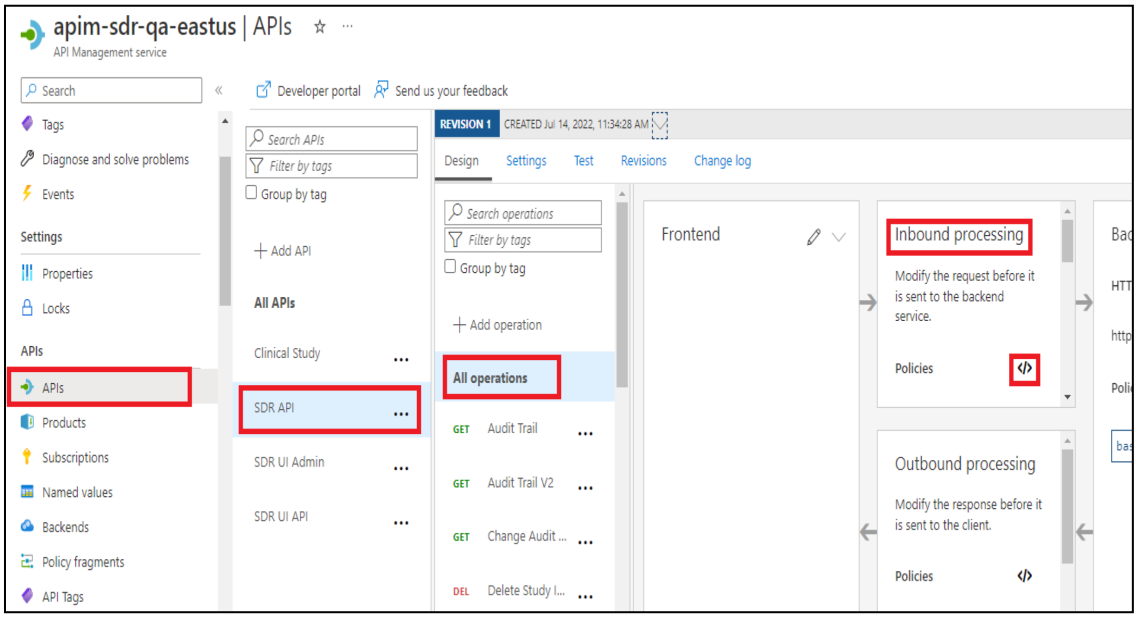
```
<policies>
    <inbound>
        <cors allow-credentials="true">
            <allowed-origins>
               <origin>Add Backend APP Service URL</origin>
                <origin>http://localhost:4200</origin>
                <origin>https://localhost:4200</origin>
                <origin>Add API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

```
</policies>
```

3.2.5 Enable Client Certificate on API Group Inbound Policies

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

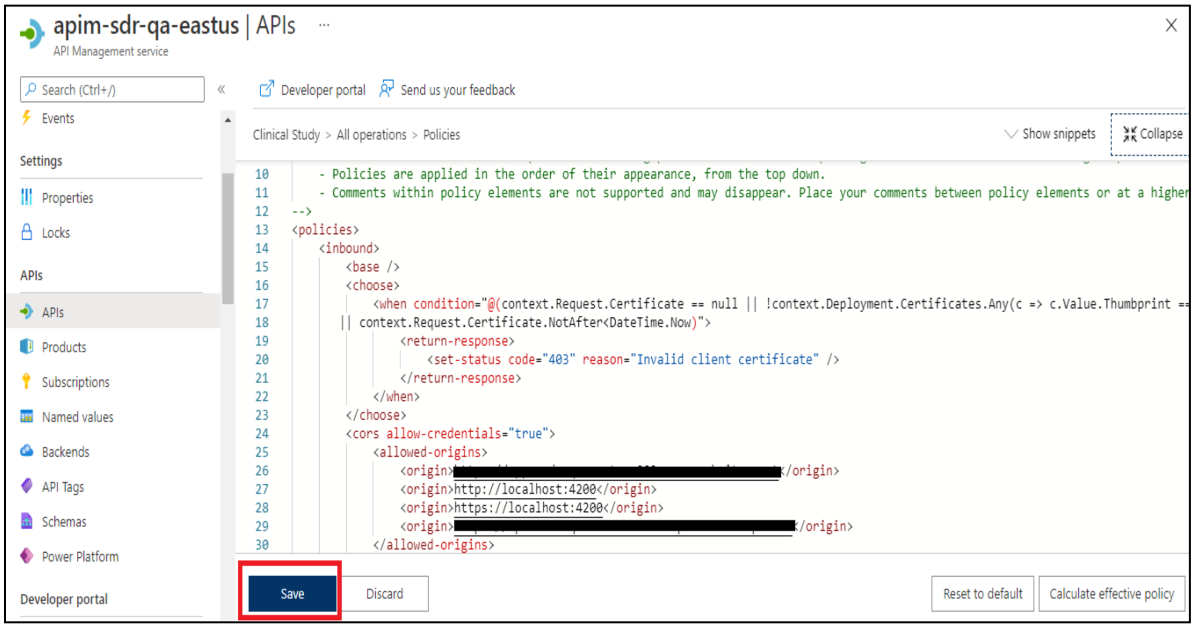
Figure 19 APIM Inbound Processing



  - 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
    return="" authHeader.AsJwt()?.Claims.GetValueOrDefault=""("email", "EmptyAuthToken");
     }" />
  <set-variable name="UserName" value="@{
     string" name = "EmptyAuthToken"
   var="" authHeader = context.Request.Headers.GetValueOrDefault
   return="" authHeader.AsJwt()?.Claims.GetValueOrDefault=""("name", "EmptyAuthToken");
     }" />
    <set-variable name="apiKey" value="@{
     string apiKey = context.Request.Headers.GetValueOrDefault("x-api-key", "EmptyApiKey");
     if((string)apiKey == ""){
         return "EmptyApiKey";
     }
     return apiKey;
     }" />
     <set-variable name="subsKeyPrimary" value="@{
     string subKey = context.Subscription?.PrimaryKey;
     return subKey;
     }" />
     <set-variable name="subsKeySecondary" value="@{
     string subKey = context.Subscription?.SecondaryKey;
     return subKey;
     }" />
 <choose>
   <when condition="@((context.Variables["EmailAddress="""]) != null)">
     <trace source="My Global APIM Policy" severity="information">
       <message>@(String.Format("{0} | {1}", context.Api.Name, context.Operation.Name))</message>
       <metadata name="EmailAddress" value="@((string)context.Variables["EmailAddress="""])" />
       <metadata name="UserName" value="@((string)context.Variables["UserName="""])" />
</trace>
   </when>
   <otherwise>
     <trace source="My Global APIM Policy" severity="information">
       <message>@(String.Format("{0} | {1}", context.Api.Name, context.Operation.Name))</message>
       <metadata name="EmailAddress" value="Not Available" />
       <metadata name="UserName" value="Not Available" />        </trace>
   </otherwise>
 </choose>
 <base />
 <choose>
   <when condition="@(context.Request.Certificate == null || !context.Deployment.Certificates.Any(c =>
c.Value.Thumbprint == context.Request.Certificate.Thumbprint)
    || context.Request.Certificate.NotAfter"
     <DateTime.Now)">
     <return-response>
       <set-status code="403" reason="Invalid client certificate" />
     </return-response>
    </when>
 </choose>
<choose>
 <when condition="@((string)context.Variables["apiKey"] != "EmptyApiKey" &&
((string)context.Variables["apiKey"] != (string)context.Variables["subsKeyPrimary"] &&
(string)context.Variables["apiKey"] != (string)context.Variables["subsKeySecondary"]))">
     <return-response>
         <set-status code="401" reason="Invalid Api Key" />
         <set-header name="content-type" exists-action="override">
             <value>application/json</value>
         </set-header>
         <set-body template="liquid">{"statusCode":"401","message":"Invalid Api-Key"}</set-body>
     </return-response>
 </when>
</choose>
 <cors allow-credentials="true">
    <allowed-origins>
     <origin>Add Backend APP Service URL</origin>
     <origin>http://localhost:4200</origin>
     <origin>https://localhost:4200</origin>
     <origin>Add API Management URL</origin>
     <origin>Add Frontend (UI) URL</origin>
     <origin>Add Developer Portal URL</origin>
    </allowed-origins>
    <allowed-methods preflight-result-max-age="300">
     <method>GET</method>
     <method>POST</method>
     <method>PATCH</method>
     <method>DELETE</method>
    </allowed-methods>
    <allowed-headers>
     <header>*</header>
    </allowed-headers>
    <expose-headers>
     <header>*</header>
    </expose-headers>
 </cors>
</inbound>
<backend>
 <base />
</backend>
<outbound>
 <base />
</outbound>
<on-error>
 <base />
</on-error>
</policies>
```

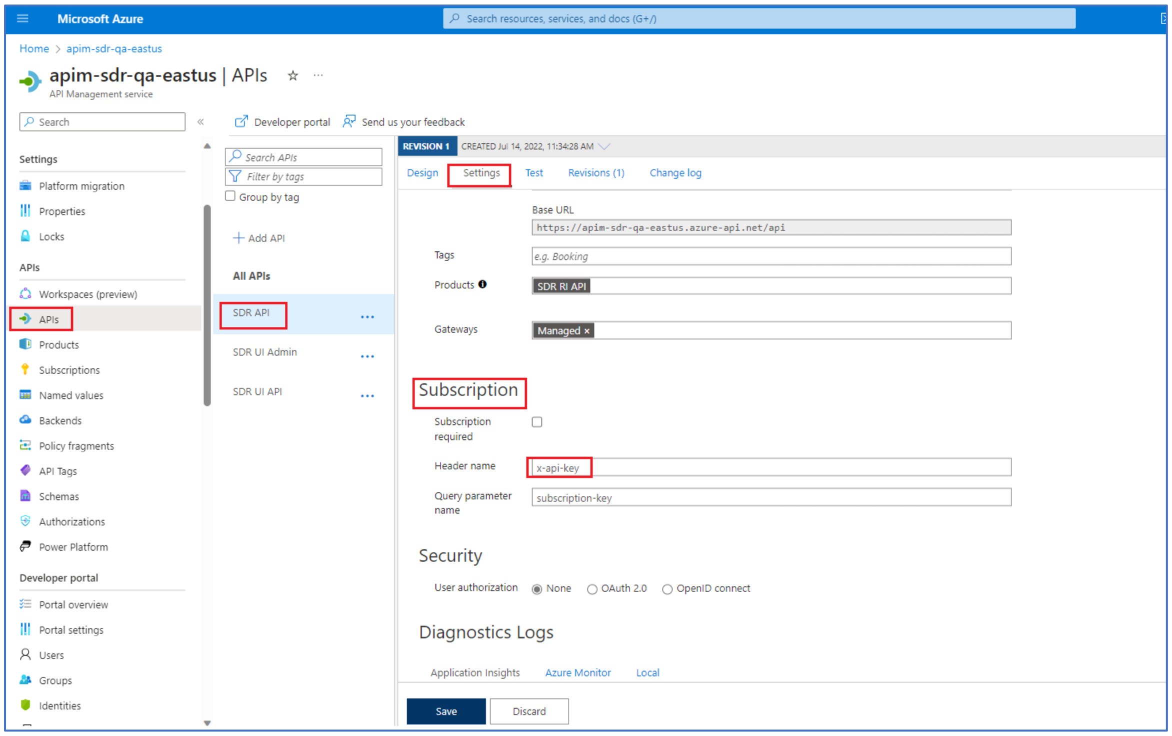
Figure 20 APIM (API Management) - Inbound Policy



```
    - Policies are applied in the order of their appearance, from the top down.
    - Comments within policy elements are not supported and may disappear. Place your comments between policy elements or at a higher
-->
<policies>
    <inbound>
        <base />
        <choose>
            <when condition="@(context.Request.Certificate == null || !context.Deployment.Certificates.Any(c => c.Value.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>[redacted]</origin>
                <origin>http://localhost:4200</origin>
                <origin>https://localhost:4200</origin>
                <origin>[redacted]</origin>
            </allowed-origins>
```

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

Figure 21 : Update the Api-Key header name



Base URL: https://apim-sdr-qa-eastus.azure-api.net/api

Tags: e.g. Booking

Products: SDR RI API

Gateways: Managed ×

Subscription

Subscription required: ☐

Header name: x-api-key

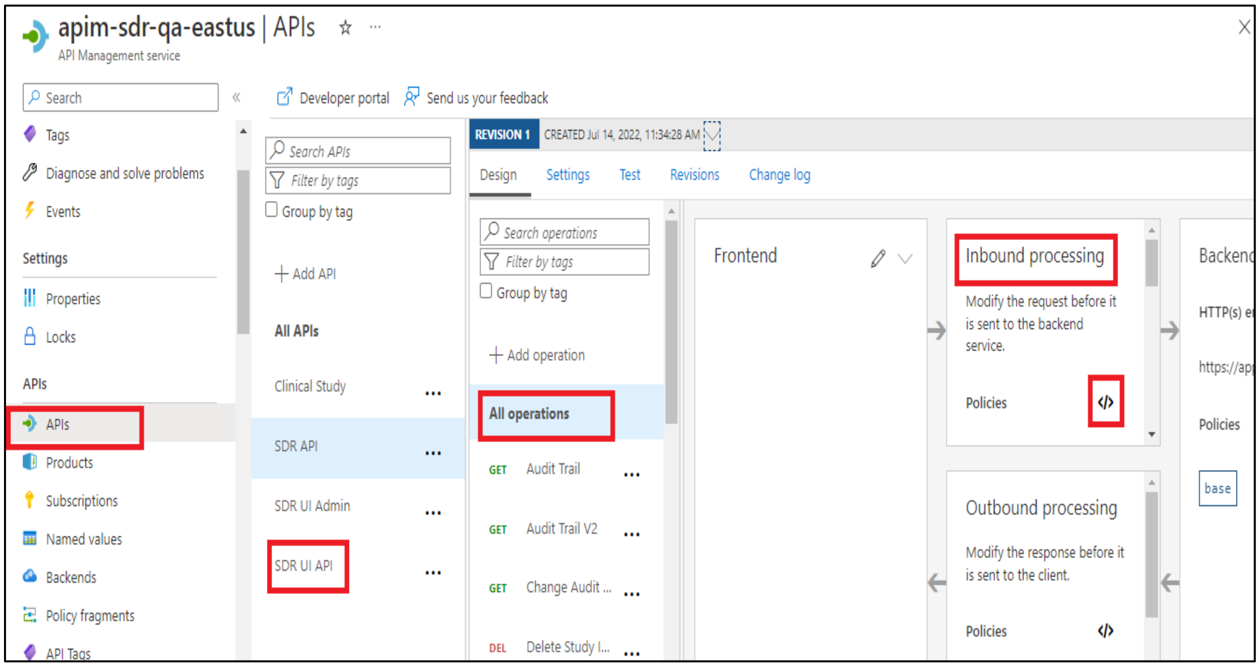
Query parameter name: subscription-key

Security

User authorization: [✓ None] OAuth 2.0 OpenID connect

Diagnostics Logs

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



- Add the policy code and make sure the inbound policy looks like below.

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

- Repeat the above steps to configure the inbound policy on SDR UI Admin endpoint.
- Now for SDR UI API and SDR UI Admin the certificate Authentication is removed.

3.2.6 Cosmos DB Updates

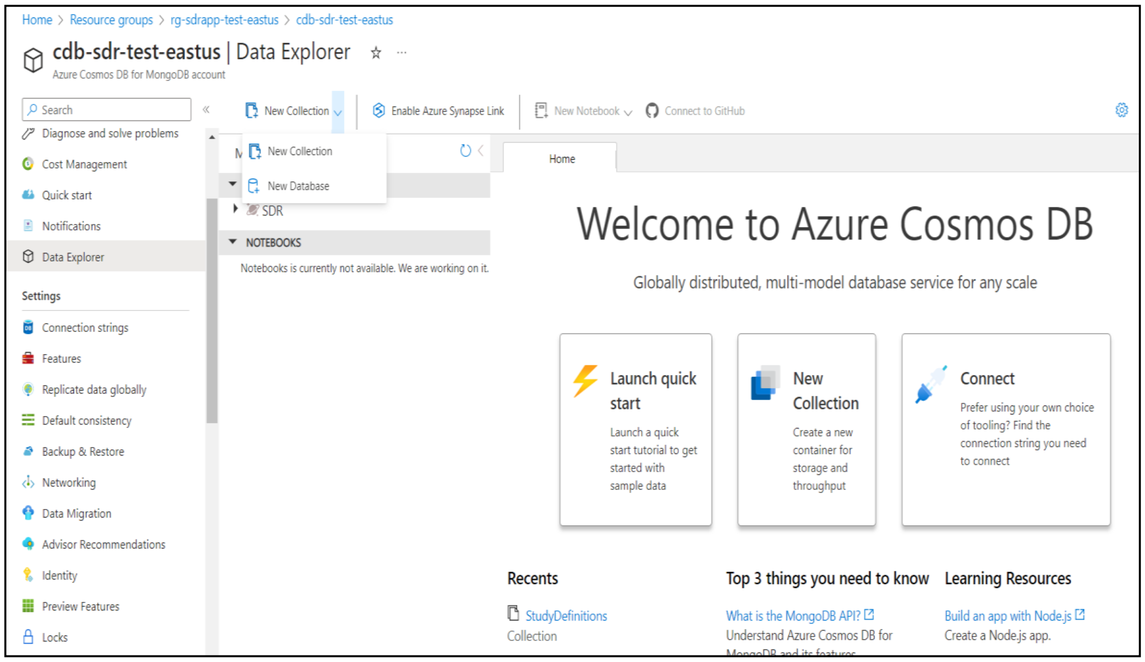
STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Go to Data Explorer and click on New Collection tab and use existing database i.e., SDR and give the Collection id (**StudyDefinitions)** then click on Ok.
5. Navigate to **StudyDefinitions** and click on settings.
6. Click on add indexing Policy and add indexes as per below table.

Index policy table for StudyDefinitions

Definition	Type
_id	Single Field
clinicalStudy.studyId	Single Field
auditTrail.entryDateTime	Single Field
clinicalStudy.studyTitle	Single Field
auditTrail.usdmVersion	Single Field

Figure 22 Adding StudyDefinitions Data Collections



Notebooks is currently not available. We are working on it.

Welcome to Azure Cosmos DB

Globally distributed, multi-model database service for any scale

Launch a quick start tutorial to get started with sample data

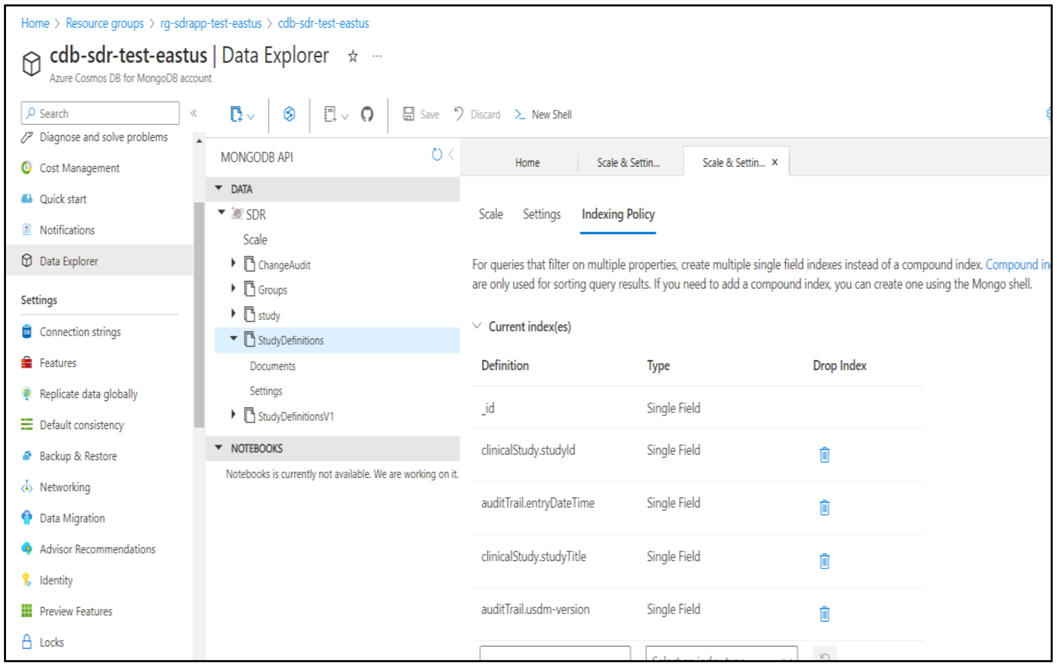
Create a new container for storage and throughput

Prefer using your own choice of tooling? Find the connection string you need to connect

Understand Azure Cosmos DB for MongoDB and its features

Create a Node.js app.

Figure 23 Adding Indexing Policy for StudyDefinitions Collection



Notebooks is currently not available. We are working on it.

For queries that filter on multiple properties, create multiple single field indexes instead of a compound index. Compound in
are only used for sorting query results. If you need to add a compound index, you can create one using the Mongo shell.

Definition	Type	Drop Index
_id	Single Field	
clinicalStudy.studyId	Single Field	🗑
auditTrail.entryDateTime	Single Field	🗑
clinicalStudy.studyTitle	Single Field	🗑
auditTrail.usdm-version	Single Field	🗑

3.2.7 Data Migration

STEPS

1. Launch Mongo shell and connect to the database using the connection string.
2. The old collection “StudyDefinitionsV1” must be updated to add “usdmVersion” parameter under AuditTrail section.
3. To add usdmVersion attribute on all documents in “StudyDefinitionsV1” collection run below command.

```
db.getCollection("StudyDefinitionsV1").updateMany({},
{$set:{"auditTrail.usdmVersion": "1.0"}}, false, true)
```

4. Run below commands to move the documents from “StudyDefinitionsV1” collection to the new common “StudyDefinitions” collection.

```
db.getCollection("StudyDefinitionsV1").aggregate([ { $merge :
"StudyDefinitions" } ])
```

5. The merge operation mentioned in previous step is supported only for MongoDB version 4.4 and above. For other lower Mongo DB versions, this activity can be done by exporting the documents from the source collection and importing into the destination collection by using any third-party IDE tools for MongoDB.
6. Run below command to update the attribute name from uuid to studyId for all USDM V1.0 documents in the new common “StudyDefinitions” collection.

```
db.getCollection("StudyDefinitions").updateMany({"auditTrail.usdmVersion":
"1.0"}, {$rename:{"clinicalStudy.uuid": "clinicalStudy.studyId"}}, false, true)
```

7. Run below command to update the parent attribute name from “clinicalStudy” to “study” for all the documents in the new common “StudyDefinitions” collection.

```
db.getCollection("StudyDefinitions").updateMany({},
{$rename:{"clinicalStudy":"study"}}, false, true)
```

3.2.8 Steps To Add Default Group

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

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

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

3.2.9 Developer Portal Configuration

APIM Developer Portal can be used to generate API-Key for SDR API access. For accessing developer portal and generating the API-Key, a few steps need to be followed. An API product needs to be created and Azure AD group access must be configured in the product. Then, the users under the group can access the developer portal and generate API-Key. Below are the steps to be followed.

Refer **DDF SDR RI Platform Setup and Deployment Guide v6.0** section 5.1 for developer portal configuration.

3.3 Additional Cloud Configurations

3.3.1 Key Vault Configurations

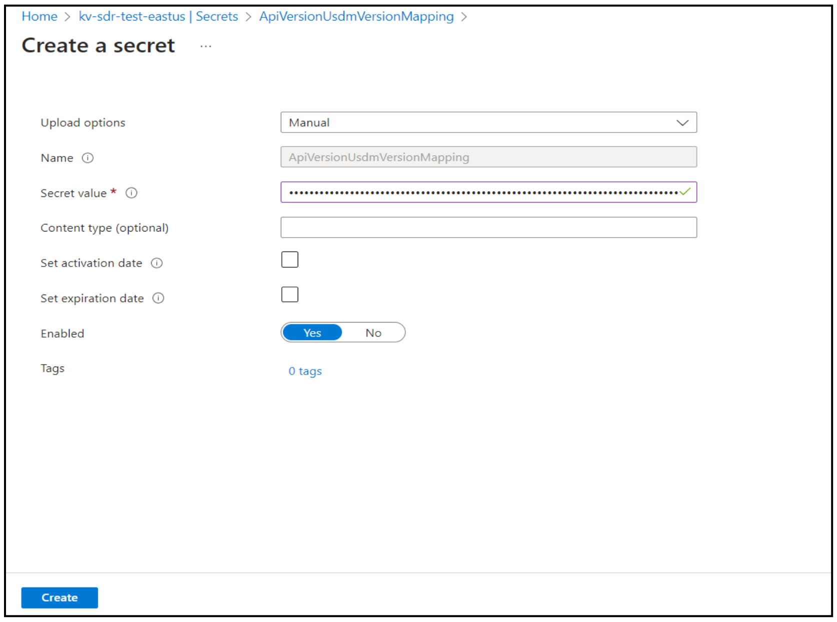
STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.

3. Select the ***sdrcore*** resource group.
4. Go to Objects tab and Click on Secret.
5. Click on Generate/Import tab and create secrets by using below table.

Name	
Env-Name	Provide a key word for the deployed environment (dev,qa,stage etc.)
ApiVersionUsdmVersionMapping	Refer to sdr-api-usdm-version-mapping.json
SdrCptMasterDataMapping	Refer to sdr-cpt-master-data-mapping.json
AzureServiceBusConnectionString	Refer to Service_Bus_Configuration Section
AzureServiceBusQueueName	Refer to Service_Bus_Configuration Section

Figure 24 Create Secret


Home > kv-sdr-test-eastus | Secrets > ApiVersionUsdmVersionMapping >

Create a secret

Upload options	Manual
Name	ApiVersionUsdmVersionMapping
Secret value *	••
Content type (optional)	
Set activation date	☐
Set expiration date	☐
Enabled	[✓ Yes] No
Tags	0 tags

3.4 Build and Deployment Updates

3.4.1 GitHub Secrets updates for deploying Function App

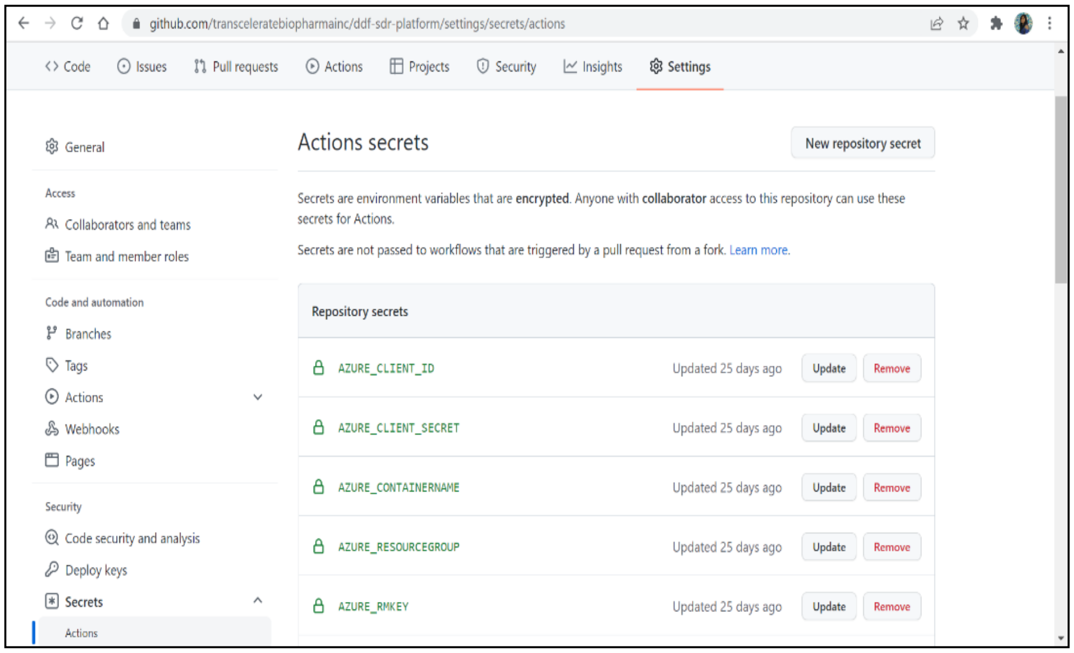
Create GitHub Secret in **ddf-sdr-api** repo required for Function App deployment.

Adding Secret in GitHub

STEPS:

1. Go to repo settings.
2. On the lower left-hand side of the screen, click on Secrets.
3. Under that click on Actions.
4. Then click on New Repository Secret.

Figure 25 GitHub Actions Secrets



Actions secrets

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

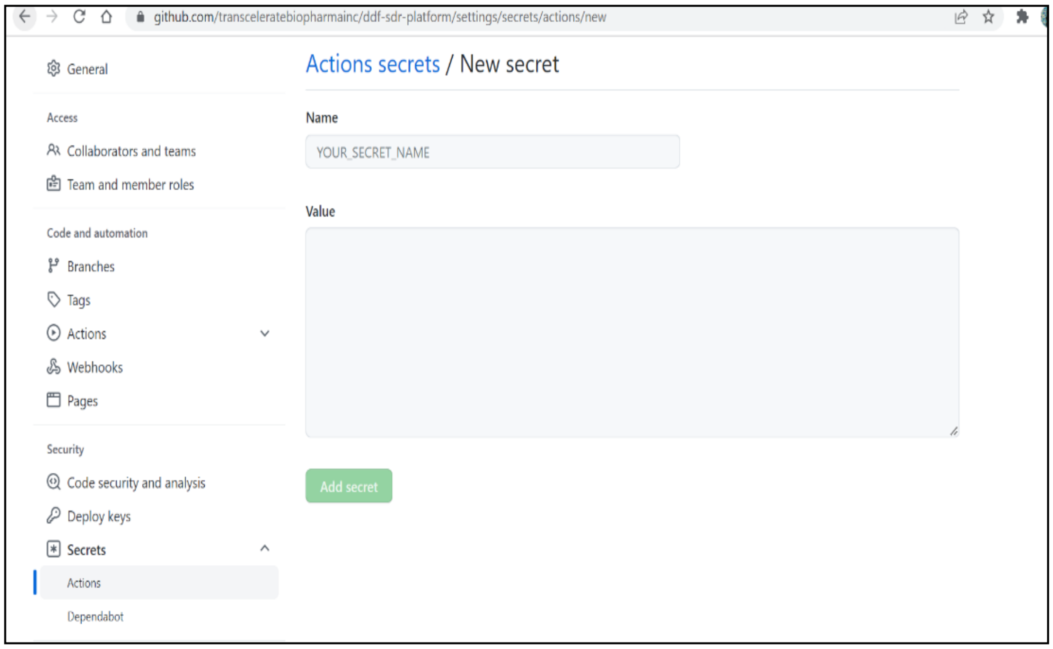
Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

Repository secrets

AZURE_CLIENT_ID	Updated 25 days ago
AZURE_CLIENT_SECRET	Updated 25 days ago
AZURE_CONTAINERNAME	Updated 25 days ago
AZURE_RESOURCEGROUP	Updated 25 days ago
AZURE_RMKEY	Updated 25 days ago

5. After clicking New Repository Secret, fill the details of the secret (name and value) in the boxes. Name – AZURE_FUNCTIONAPP_NAME and Value – name of function app resource created in Azure e.g.: funapp-sdr-dev-eastus.

Figure 26 Add new GitHub Action Secret


github.com/transceleratebiopharmainc/ddf-sdr-platform/settings/secrets/actions/new

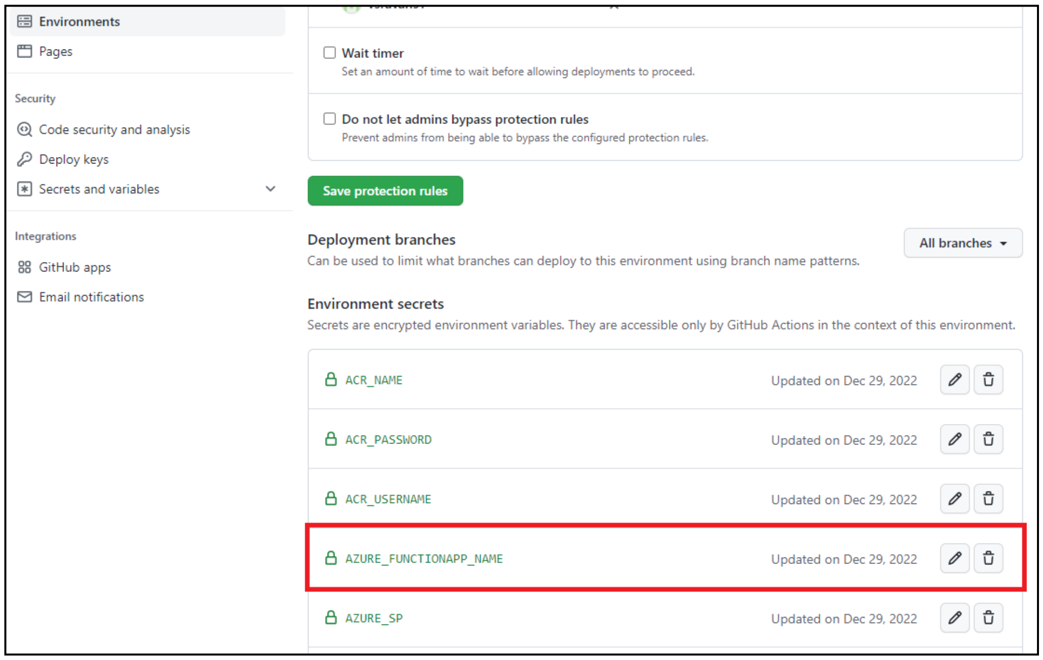
Actions secrets / New secret

Name

YOUR_SECRET_NAME

Value

Figure 27 Function App GitHub Secret



- [ ] Wait timer
  Set an amount of time to wait before allowing deployments to proceed.
- [ ] Do not let admins bypass protection rules
  Prevent admins from being able to bypass the configured protection rules.

Deployment branches

Can be used to limit what branches can deploy to this environment using branch name patterns.

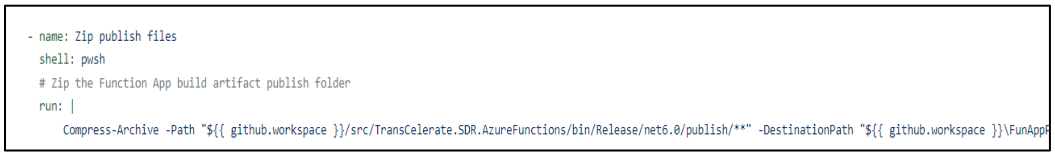
Environment secrets

Secrets are encrypted environment variables. They are accessible only by GitHub Actions in the context of this environment.

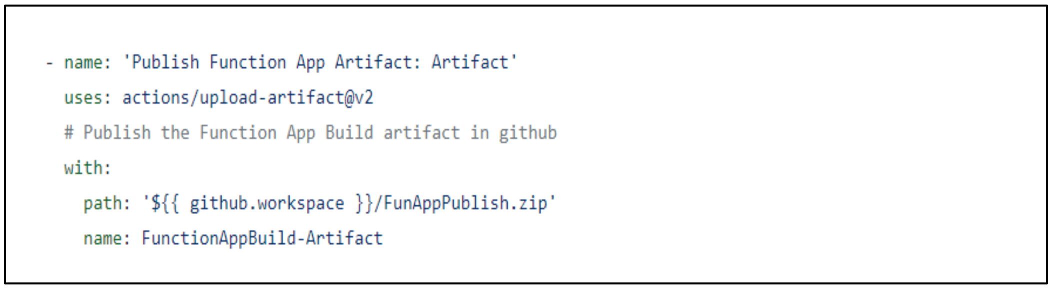
ACR_NAME	Updated on Dec 29, 2022
ACR_PASSWORD	Updated on Dec 29, 2022
ACR_USERNAME	Updated on Dec 29, 2022
AZURE_FUNCTIONAPP_NAME	Updated on Dec 29, 2022
AZURE_SP	Updated on Dec 29, 2022

6. Update GitHub Action yml file in ddf-sdr-api repo. A new task has been created to publish function App Artifacts. Please refer the below screen shot.

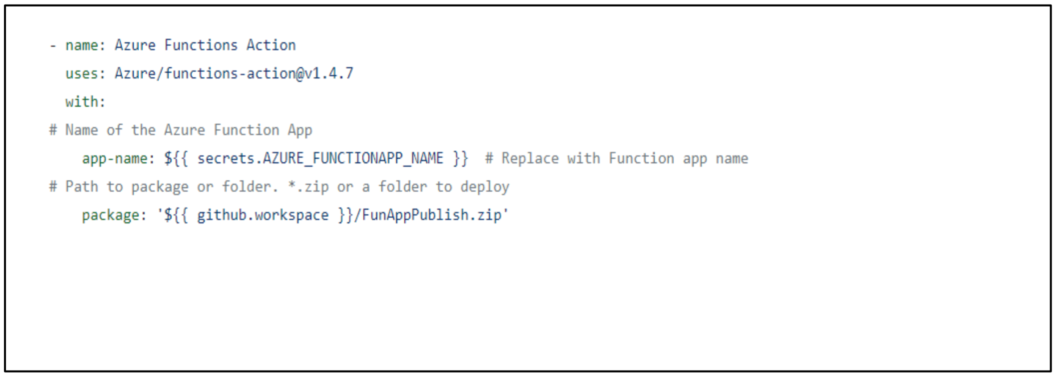
Figure 28 Adding a new task in yml file to Publish Function App Build Artifact



```
- name: Zip publish files
  shell: pwsh
  # Zip the Function App build artifact publish folder
  run: |
      Compress-Archive -Path "${{ github.workspace }}/src/TransCelerate.SDR.AzureFunctions/bin/Release/net6.0/publish/**" -DestinationPath "${{ github.workspace }}\FunAppP
```



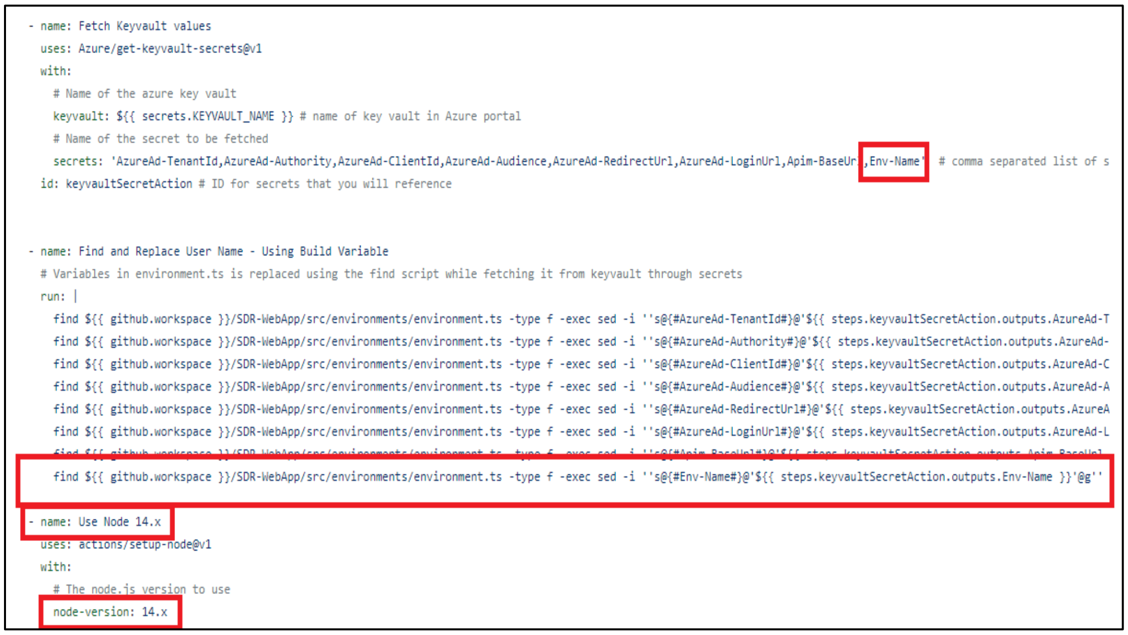
```
- name: 'Publish Function App Artifact: Artifact'
  uses: actions/upload-artifact@v2
  # Publish the Function App Build artifact in github
  with:
    path: '${{ github.workspace }}/FunAppPublish.zip'
    name: FunctionAppBuild-Artifact
```



```
- name: Azure Functions Action
  uses: Azure/functions-action@v1.4.7
  with:
# Name of the Azure Function App
    app-name: ${{ secrets.AZURE_FUNCTIONAPP_NAME }}  # Replace with Function app name
# Path to package or folder. *.zip or a folder to deploy
    package: '${{ github.workspace }}/FunAppPublish.zip'
```

- **ddf-sdr-ui repo YAML script Changes**

  1. A new Key vault variable is added in yml file to display the Environment in the System Usage report.
  2. Additionally, the node.js version from 12.x to 14.x. Please refer the below screen shots



```
- name: Fetch Keyvault values
  uses: Azure/get-keyvault-secrets@v1
  with:
    # Name of the azure key vault
    keyvault: ${{ secrets.KEYVAULT_NAME }} # name of key vault in Azure portal
    # Name of the secret to be fetched
    secrets: 'AzureAd-TenantId,AzureAd-Authority,AzureAd-ClientId,AzureAd-Audience,AzureAd-RedirectUrl,AzureAd-LoginUrl,Apim-BaseUr ,Env-Name'  # comma separated list of s
  id: keyvaultSecretAction # ID for secrets that you will reference


- name: Find and Replace User Name - Using Build Variable
  # Variables in environment.ts is replaced using the find script while fetching it from keyvault through secrets
  run: |
    find ${{ github.workspace }}/SDR-WebApp/src/environments/environment.ts -type f -exec sed -i ''s@{#AzureAd-TenantId#}@'${{ steps.keyvaultSecretAction.outputs.AzureAd-T
    find ${{ github.workspace }}/SDR-WebApp/src/environments/environment.ts -type f -exec sed -i ''s@{#AzureAd-Authority#}@'${{ steps.keyvaultSecretAction.outputs.AzureAd-
    find ${{ github.workspace }}/SDR-WebApp/src/environments/environment.ts -type f -exec sed -i ''s@{#AzureAd-ClientId#}@'${{ steps.keyvaultSecretAction.outputs.AzureAd-C
    find ${{ github.workspace }}/SDR-WebApp/src/environments/environment.ts -type f -exec sed -i ''s@{#AzureAd-Audience#}@'${{ steps.keyvaultSecretAction.outputs.AzureAd-A
    find ${{ github.workspace }}/SDR-WebApp/src/environments/environment.ts -type f -exec sed -i ''s@{#AzureAd-RedirectUrl#}@'${{ steps.keyvaultSecretAction.outputs.AzureA
    find ${{ github.workspace }}/SDR-WebApp/src/environments/environment.ts -type f -exec sed -i ''s@{#AzureAd-LoginUrl#}@'${{ steps.keyvaultSecretAction.outputs.AzureAd-L
    find ${{ github.workspace }}/SDR-WebApp/src/environments/environment.ts -type f -exec sed -i ''s@{#Apim-BaseUrl#}@'${{ steps.keyvaultSecretAction.outputs.Apim-BaseUrl
    find ${{ github.workspace }}/SDR-WebApp/src/environments/environment.ts -type f -exec sed -i ''s@{#Env-Name#}@'${{ steps.keyvaultSecretAction.outputs.Env-Name }}'@g''

- name: Use Node 14.x
  uses: actions/setup-node@v1
  with:
    # The node.js version to use
    node-version: 14.x
```

Figure 29 Adding new key vault value in yml file

3.4.2 Code Deployment

To enable containerized deployment of SDR application, please skip this step and refer to section 4. Otherwise, the UI and API code deployment to respective App services can be done using the below updated scripts.

[ddf-sdr-api/release.yml at develop · transcelerate/ddf-sdr-api (github.com)]

[ddf-sdr-ui/release.yml at develop · transcelerate/ddf-sdr-ui (github.com)]

4 Containerized Deployment

The UI and API builds have been containerized as a part of SDR Release V2.0. The same has impacted the App Services on which the SDR Application is hosted. This section details the steps to enable containerized build and deployment.

This is an optional step. To continue with existing way of deployment of SDR use current App Service and Service Plan configuration and B&D scripts for UI and API for deployment as mentioned in section 3.4.2.

4.1 Recreate App Service Plans and App Services

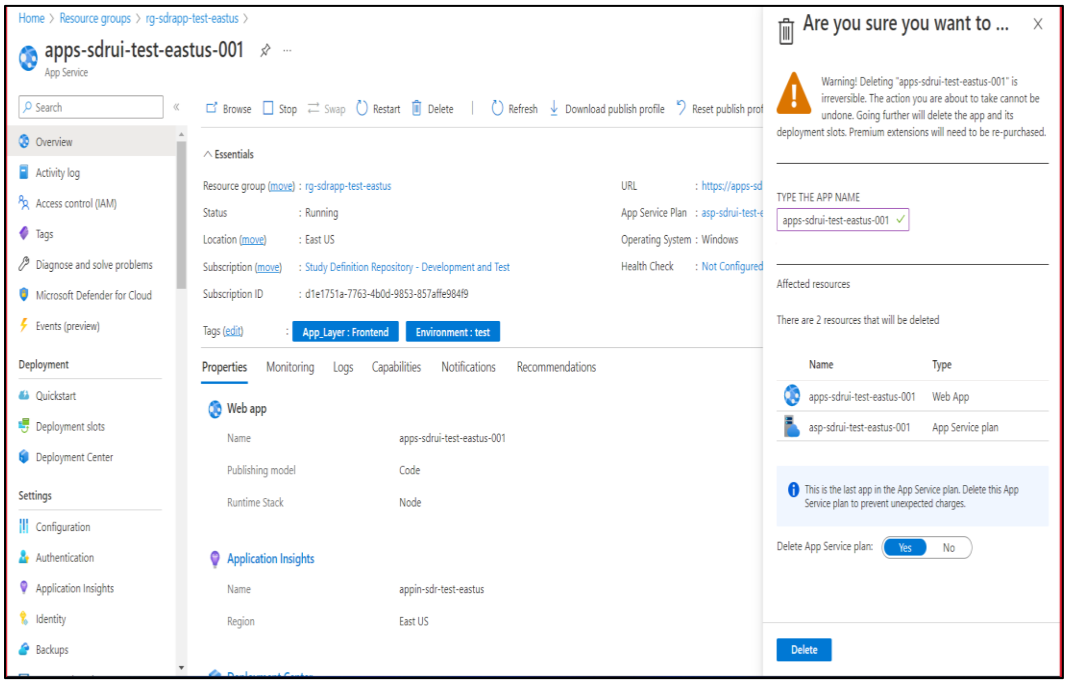
4.1.1 Destroy existing UI app service Plan and App Service

STEPS

1. Login to Azure Portal
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.

4. Select the UI App Service and Click on Delete from the Overview Page and delete both App service and App Service Plans

Figure 30 Destroy existing UI app service Plan and App Service



apps-sdrui-test-eastus-001
App Service

Essentials

Resource group (move) : rg-sdrapp-test-eastus
Status : Running
Location (move) : East US
Subscription (move) : Study Definition Repository - Development and Test
Subscription ID : d1e1751a-7763-4b0d-9853-857affe984f9
Tags (edit) : App_Layer : Frontend Environment : test

URL : https://apps-sd
App Service Plan : asp-sdrui-test-e
Operating System : Windows
Health Check : Not Configured

Web app
Name apps-sdrui-test-eastus-001
Publishing model Code
Runtime Stack Node

Application Insights
Name appin-sdr-test-eastus
Region East US

Are you sure you want to ...

Warning! Deleting "apps-sdrui-test-eastus-001" is irreversible. The action you are about to take cannot be undone. Going further will delete the app and its deployment slots. Premium extensions will need to be re-purchased.

TYPE THE APP NAME
apps-sdrui-test-eastus-001 ✓

Affected resources

There are 2 resources that will be deleted

Name	Type
apps-sdrui-test-eastus-001	Web App
asp-sdrui-test-eastus-001	App Service plan

This is the last app in the App Service plan. Delete this App Service plan to prevent unexpected charges.

Delete App Service plan: [✓ Yes] No

4.1.2 Recreate UI app Service Plan

STEPS

1 Login to Azure Portal
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Refer the LLD document and create an App Service plan for SDR UI

4.1.3 Recreate UI App Service

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Refer the LLD document and create an App Service for SDR UI.

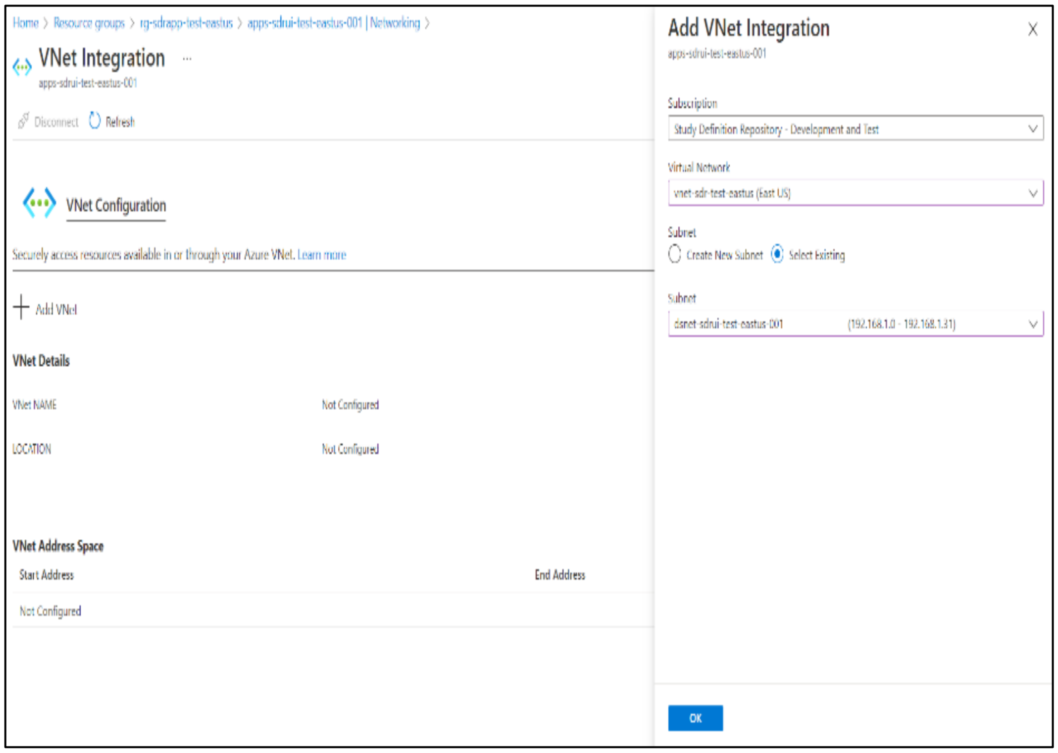
4.1.4 SDR UI App Service Configuration

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Select the UI App Service and navigate to networking tab.

5. Click on VNet Integration and add VNet and subnet then click on Ok.
6. Go to Monitoring Tab and select Diagnostic Setting.
7. Click on add Diagnostic Settings and give the Diagnostic Setting Name.
8. Select Logs and Metrics and check “Send to Log Analytics Workspace”
9. Save the settings.

Figure 31 SDR UI Network Configuration


Home > Resource groups > rg-sdrapp-test-eastus > apps-sdrui-test-eastus-001 | Networking >
VNet Integration ...
apps-sdrui-test-eastus-001
Disconnect
Refresh
VNet Configuration
Securely access resources available in or through your Azure VNet. Learn more
+ Add VNet
VNet Details
VNet Address Space
Add VNet Integration
apps-sdrui-test-eastus-001
OK

VNet NAME	Not Configured
LOCATION	Not Configured

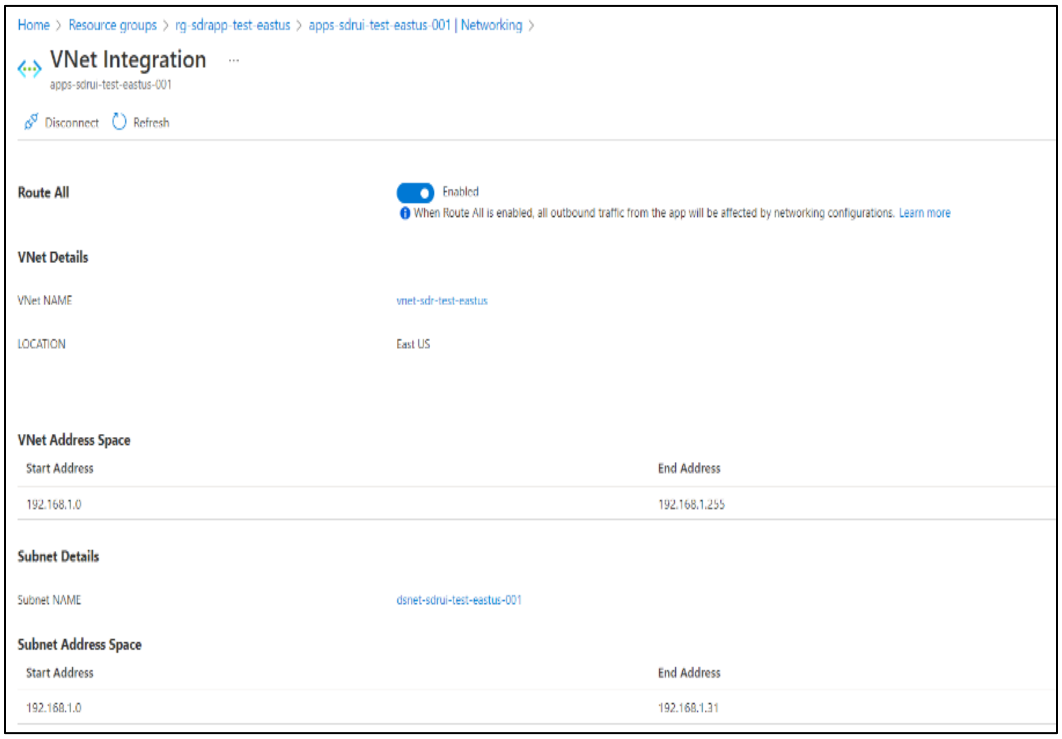
Start Address	End Address
Not Configured	

Subscription: Study Definition Repository - Development and Test

Virtual Network: vnet-sdr-test-eastus (East US)

Subnet: Create New Subnet [✓ Select Existing]

Subnet: dsnet-sdrui-test-eastus-001 (192.168.1.0 - 192.168.1.31)



Home > Resource groups > rg-sdrapp-test-eastus > apps-sdrui-test-eastus-001 | Networking >

VNet Integration

apps-sdrui-test-eastus-001

Disconnect Refresh

Route All	Enabled. When Route All is enabled, all outbound traffic from the app will be affected by networking configurations. Learn more

VNet Details

VNet NAME	vnet-sdr-test-eastus
LOCATION	East US

VNet Address Space

Start Address	End Address
192.168.1.0	192.168.1.255

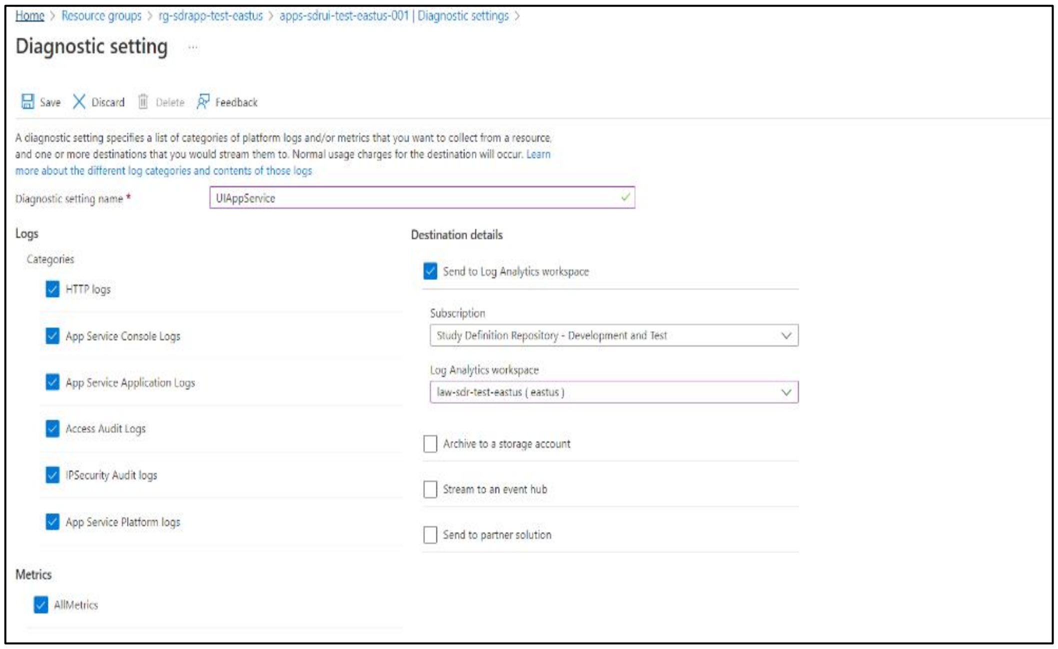
Subnet Details

Subnet NAME	dsnet-sdrui-test-eastus-001

Subnet Address Space

Start Address	End Address
192.168.1.0	192.168.1.31

Figure 32 SDR UI Diagnostic Configuration



Home > Resource groups > rg-sdrapp-test-eastus > apps-sdrui-test-eastus-001 | Diagnostic settings >

Diagnostic setting

Save Discard Delete Feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. Learn more about the different log categories and contents of those logs

Diagnostic setting name * UIAppService

Logs

Categories

- ✓ HTTP logs
- ✓ App Service Console Logs
- ✓ App Service Application Logs
- ✓ Access Audit Logs
- ✓ IPSecurity Audit logs
- ✓ App Service Platform logs

Metrics

- ✓ AllMetrics

Destination details

- ✓ Send to Log Analytics workspace
  - Subscription: Study Definition Repository - Development and Test
  - Log Analytics workspace: law-sdr-test-eastus (eastus)
- ☐ Archive to a storage account
- ☐ Stream to an event hub
- ☐ Send to partner solution

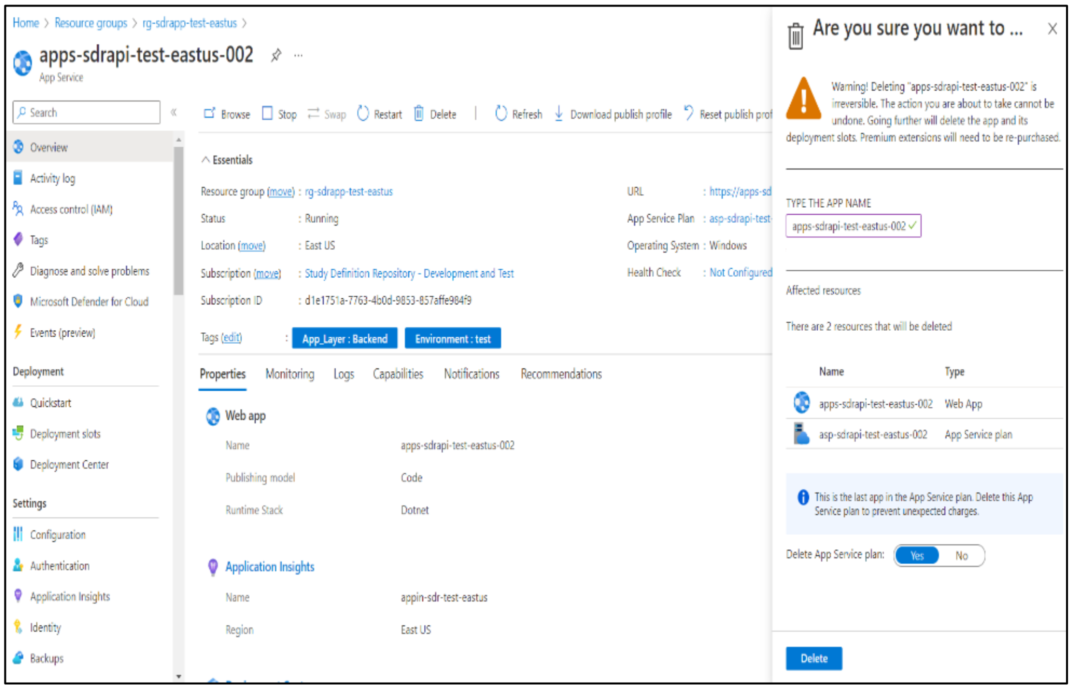
4.1.5 Destroy existing API app service Plan and App Service

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.

3. Select the ***sdrapp*** resource group.
4. Select the API App Service and Click on Delete from the Overview page and delete both App Service and App Service Plan

Figure 33 SDR API App Service Details



Home > Resource groups > rg-sdrapp-test-eastus >

apps-sdrapi-test-eastus-002
App Service

Essentials

Resource group (move)	: rg-sdrapp-test-eastus	URL	: https://apps-sd
Status	: Running	App Service Plan	: asp-sdrapi-test-
Location (move)	: East US	Operating System	: Windows
Subscription (move)	: Study Definition Repository - Development and Test	Health Check	: Not Configured
Subscription ID	: d1e1751a-7763-4b0d-9853-857affe984f9		
Tags (edit)	: App_Layer : Backend Environment : test		

Web app

Name	apps-sdrapi-test-eastus-002
Publishing model	Code
Runtime Stack	Dotnet

Application Insights

Name	appin-sdr-test-eastus
Region	East US

Are you sure you want to ...

Warning! Deleting "apps-sdrapi-test-eastus-002" is irreversible. The action you are about to take cannot be undone. Going further will delete the app and its deployment slots. Premium extensions will need to be re-purchased.

TYPE THE APP NAME
apps-sdrapi-test-eastus-002 ✓

Affected resources

There are 2 resources that will be deleted

Name	Type
apps-sdrapi-test-eastus-002	Web App
asp-sdrapi-test-eastus-002	App Service plan

This is the last app in the App Service plan. Delete this App Service plan to prevent unexpected charges.

Delete App Service plan: [✓ Yes] No

Delete

4.1.6 Recreate API App Service Plan

STEPS

1. Login to Azure Portal
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Refer the LLD document and create an App Service plan for SDR API

4.1.7 Create API App Service

STEPS

1 Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Refer the LLD document and create an App Service for SDR API.

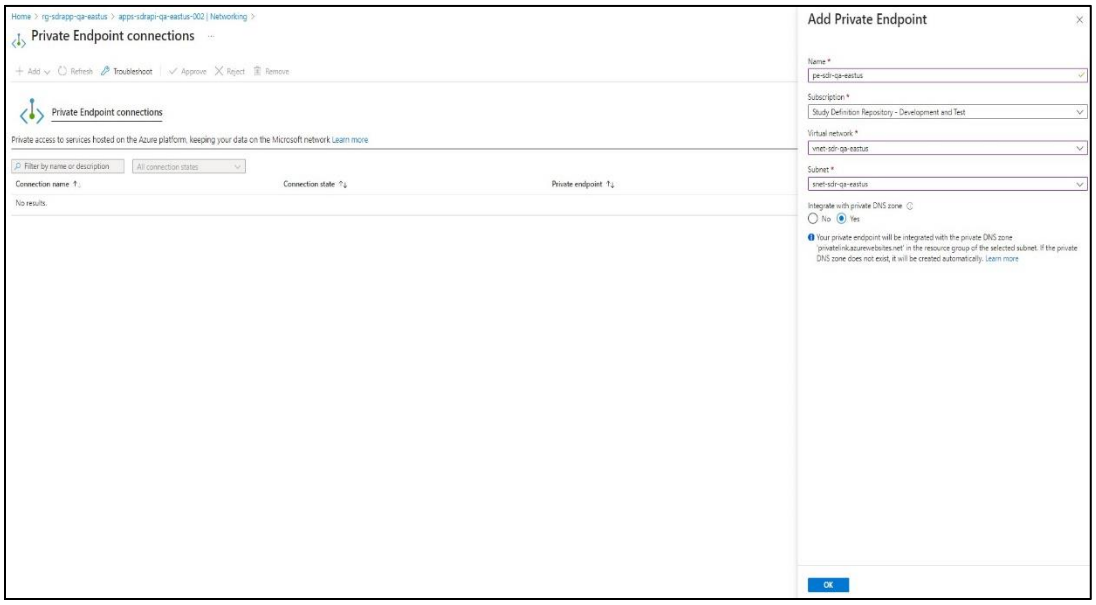
4.1.8 SDR API App Service Configuration

STEPS

1. Login to Azure Portal.

2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Select the API App Service and navigate to networking tab.
5. Click on private endpoints and add a new private endpoint with same VNet/Subnet configuration as of API Management resource to enable backend connectivity. Also, enable private DNS zone integration when creating the private endpoint.
6. Go to Monitoring Tab and select Diagnostic Setting.
7. Click on add Diagnostic Settings and give the Diagnostic Setting Name.
8. Select Logs and Metrics and send to Log Analytics Workspace
9. Save the settings.
10. Go to Identity tab and enable System assigned managed identity and save the settings.
11. To enable KeyVault access for API app service, Navigate to **sdrcore** resource group.
12. Select Azure Key Vault and navigate to Access policies tab.
13. Click on Create and select **Secret Management** option from the **Configure a template** dropdown.
14. Select all applicable **secret management operations**
15. Click on Next and search for API app name and select API App Service (E.g., apps-sdrapi-qa-estus-002).
16. Click on Next and click on Create.

Figure 34 SDR API Private endpoint Configuration


Home > rg-sdrapp-qa-eastus > apps-sdrapi-qa-eastus-002 | Networking >
Private Endpoint connections
Add
Refresh
Troubleshoot
Approve
Reject
Remove
Private Endpoint connections
Filter by name or description
All connection states
OK

Private access to services hosted on the Azure platform, keeping your data on the Microsoft network Learn more

Connection name ↑↓	Connection state ↑↓	Private endpoint ↑↓
No results.		

Add Private Endpoint

Name *: pe-sdr-qa-eastus

Subscription *: Study Definition Repository - Development and Test

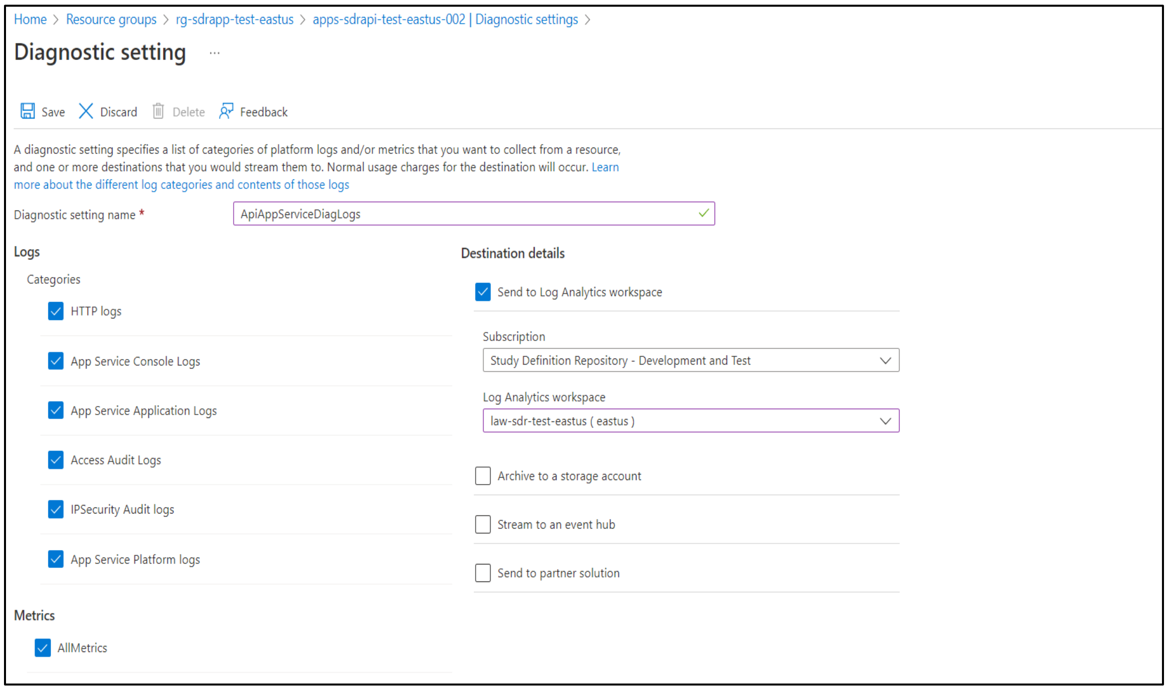
Virtual network *: vnet-sdr-qa-eastus

Subnet *: snet-sdr-qa-eastus

Integrate with private DNS zone: No [✓ Yes]

Your private endpoint will be integrated with the private DNS zone 'privatelink.azurewebsites.net' in the resource group of the selected subnet. If the private DNS zone does not exist, it will be created automatically. Learn more

Figure 35 SDR API Diagnostic Configuration



Home > Resource groups > rg-sdrapp-test-eastus > apps-sdrapi-test-eastus-002 | Diagnostic settings >

Diagnostic setting ...

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. Learn more about the different log categories and contents of those logs

Diagnostic setting name * ApiAppServiceDiagLogs

Logs

Categories

- ✓ HTTP logs
- ✓ App Service Console Logs
- ✓ App Service Application Logs
- ✓ Access Audit Logs
- ✓ IPSecurity Audit logs
- ✓ App Service Platform logs

Metrics

- ✓ AllMetrics

Destination details

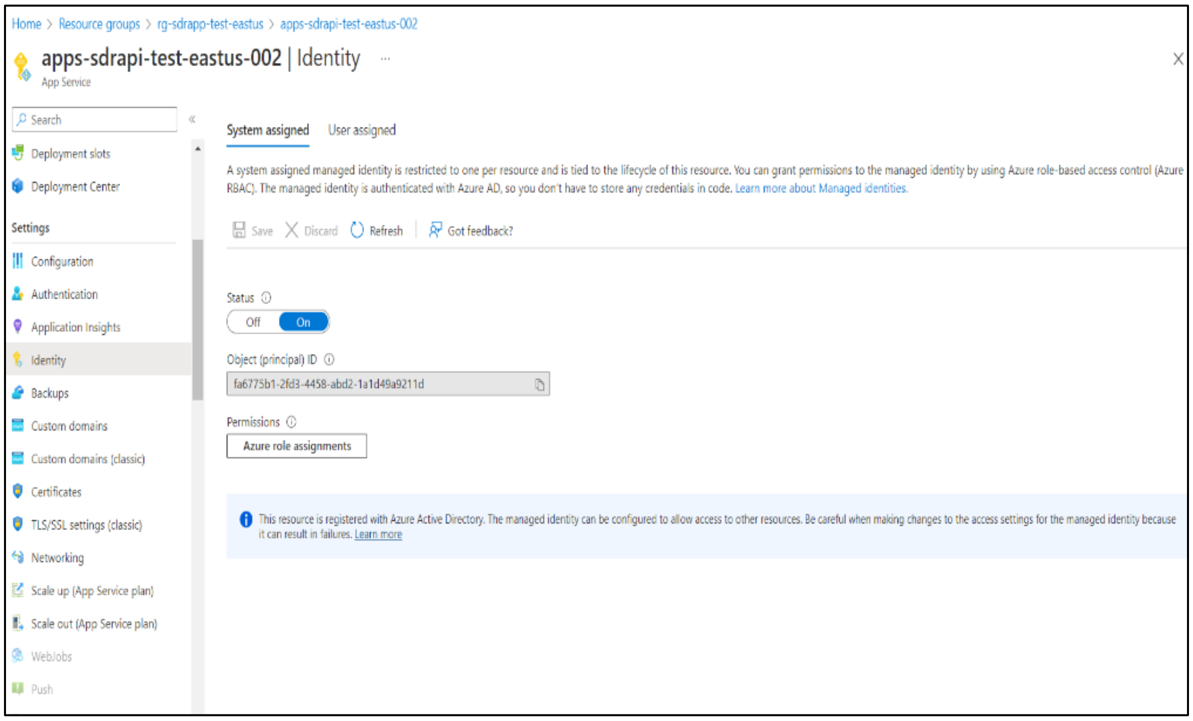
- ✓ Send to Log Analytics workspace

Subscription: Study Definition Repository - Development and Test

Log Analytics workspace: law-sdr-test-eastus (eastus)

- ☐ Archive to a storage account
- ☐ Stream to an event hub
- ☐ Send to partner solution

Figure 36 SDR API Identity Configuration



Home > Resource groups > rg-sdrapp-test-eastus > apps-sdrapi-test-eastus-002

apps-sdrapi-test-eastus-002 | Identity

App Service

System assigned | User assigned

A system assigned managed identity is restricted to one per resource and is tied to the lifecycle of this resource. You can grant permissions to the managed identity by using Azure role-based access control (Azure RBAC). The managed identity is authenticated with Azure AD, so you don't have to store any credentials in code. Learn more about Managed identities.

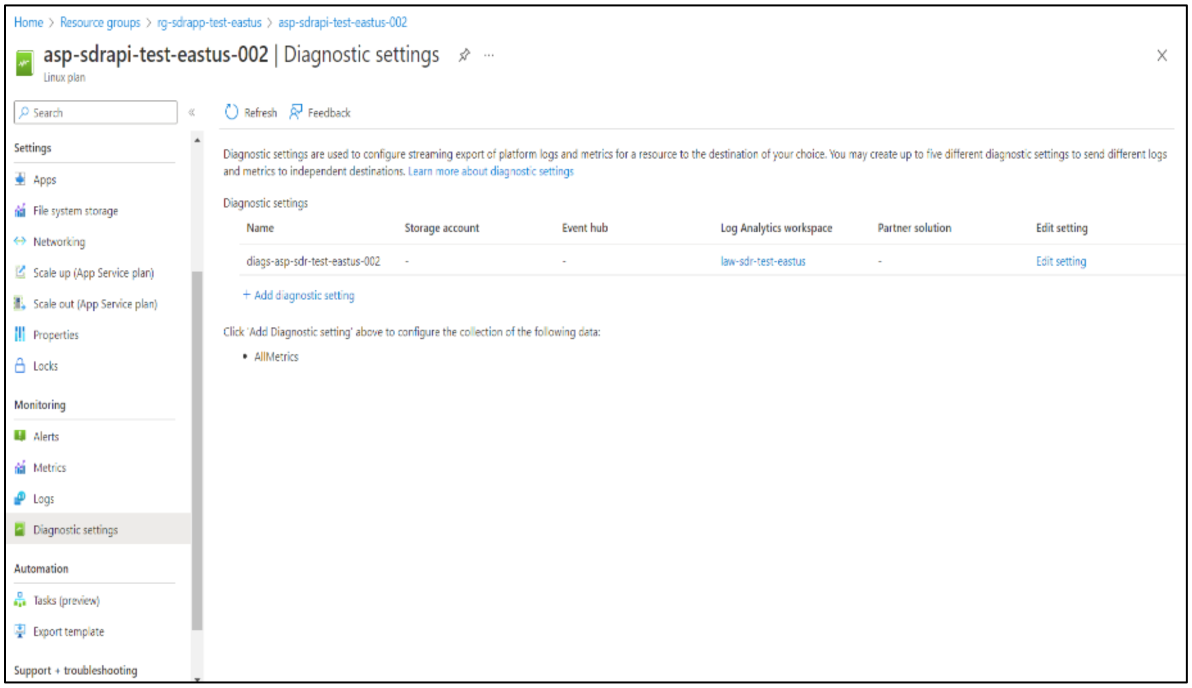
Status: On

Object (principal) ID: fa6775b1-2fd3-4458-abd2-1a1d49a9211d

Permissions: Azure role assignments

This resource is registered with Azure Active Directory. The managed identity can be configured to allow access to other resources. Be careful when making changes to the access settings for the managed identity because it can result in failures. Learn more

Figure 37 SDR API Diagnostic Configuration



Home > Resource groups > rg-sdrapp-test-eastus > asp-sdrapi-test-eastus-002

asp-sdrapi-test-eastus-002 | Diagnostic settings

Linux plan

Diagnostic settings are used to configure streaming export of platform logs and metrics for a resource to the destination of your choice. You may create up to five different diagnostic settings to send different logs and metrics to independent destinations. Learn more about diagnostic settings

Diagnostic settings

Name	Storage account	Event hub	Log Analytics workspace	Partner solution	Edit setting
diags-asp-sdr-test-eastus-002	-	-	law-sdr-test-eastus	-	Edit setting

+ Add diagnostic setting

Click 'Add Diagnostic setting' above to configure the collection of the following data:

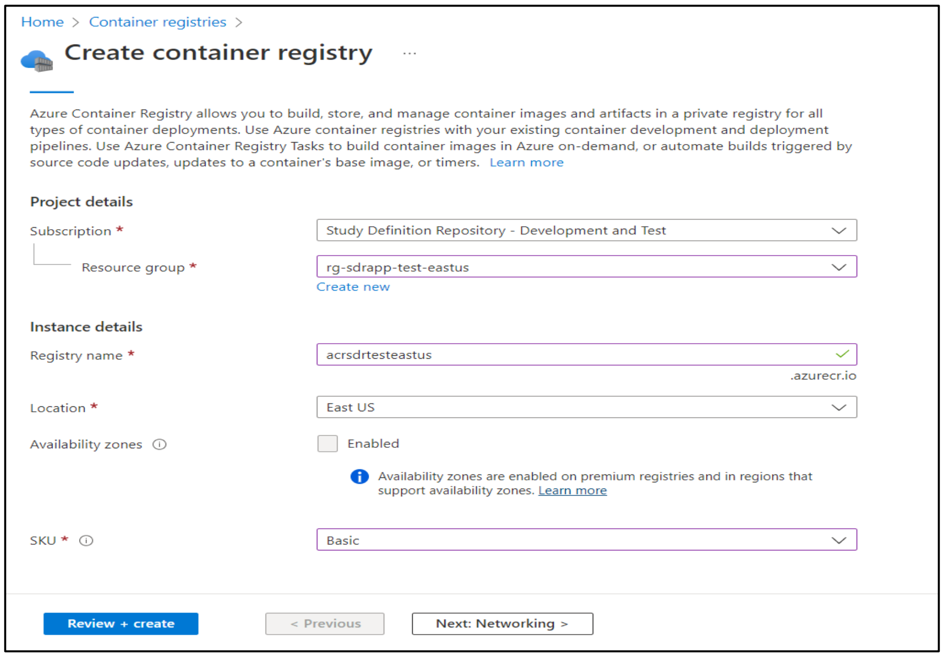
- AllMetrics

4.1.9 Create Azure Container Registry

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Refer the LLD document and create an Azure Container Registry
   - Basics Details (Subscription, Resource Group)
   - Instance Details (Registry name, Location, Availability Zones, SKU)
   - Networking
   - Encryption
   - Tags

Figure 38 Container Registry Basics Details



Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments. Use Azure container registries with your existing container development and deployment pipelines. Use Azure Container Registry Tasks to build container images in Azure on-demand, or automate builds triggered by source code updates, updates to a container's base image, or timers. Learn more

Project details

Subscription *	Study Definition Repository - Development and Test
Resource group *	rg-sdrapp-test-eastus

Create new

Instance details

Registry name *	acrsdrtesteastus .azurecr.io
Location *	East US
Availability zones	Enabled
SKU *	Basic

Availability zones are enabled on premium registries and in regions that support availability zones. Learn more

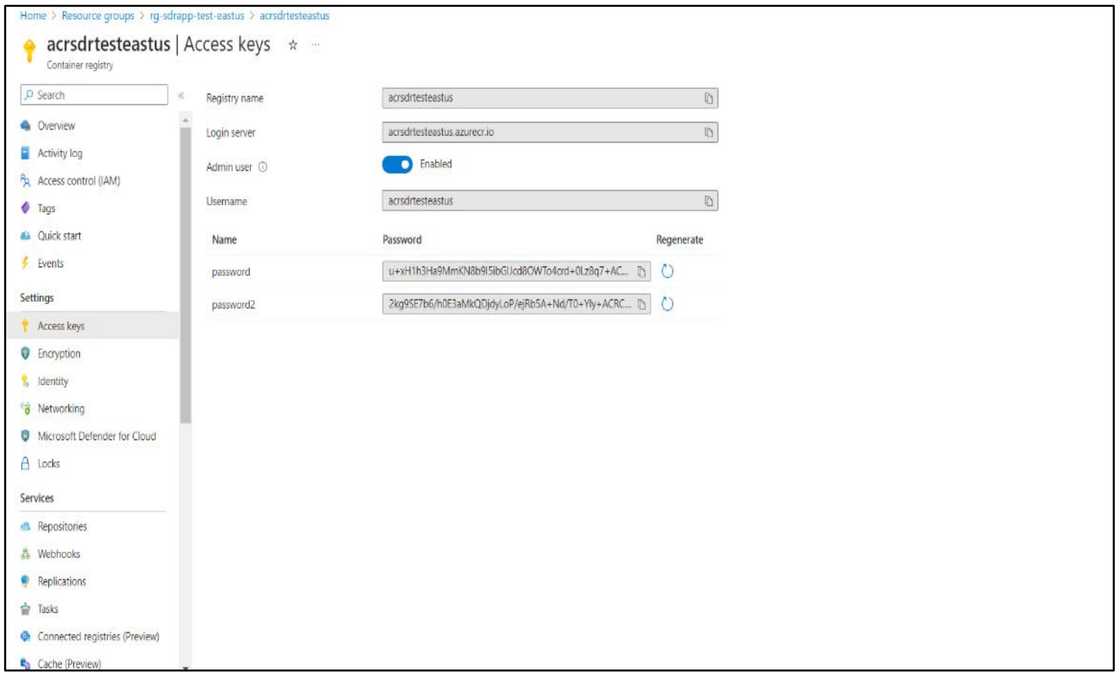
4.1.10 Container Registry Configuration

STEPS

1. Login to Azure Portal.
2. Click on the Resource Groups tab.
3. Select the ***sdrapp*** resource group.
4. Select Container Registry and navigate to Access key and enable Admin user setting.
5. Capture Username, Password and Login Server URL.

Note: Please make a note of Username, Password and Login Server URL values required to be added in UI & API Application Settings.

Figure 39 Enabling Admin User Settings



Registry name: acrsdrtesteastus

Login server: acrsdrtesteastus.azurecr.io

Admin user: Enabled

Username: acrsdrtesteastus

Name	Password	Regenerate
password	u+xH1h3Ha9MmKN8b9I5ibGIJcd8OWTo4crd+0Lz8q7+AC...	
password2	2kg9SE7b6/h0E3aMkQDjdyLoP/ejRb5A+Nd/T0+Yly+ACRC...	

4.2 GitHub Actions/YAML/Build & Deploy Script Updates

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.

Below are the steps to Containerize an application and deploy via ci/cd pipeline.

- Create a docker file and add it in the root folder of the GitHub repo of source code.
- Create an Azure Container Registry to store container images.
- Create GitHub Actions workflow to build the container and deploy it to App service.

The "**Dockerfile**" file is used by the docker build command to create a container image.

By selecting Create a new project and Docker is enabled, the "Dockerfile" and ".dockerignore" files are automatically generated for an ASP.NET Core Web app in Visual Studio.

4.2.1 GitHub Actions to deploy container image to UI App Service

The Azure Web Deploy action automates the workflow to deploy custom container image to app service.

PRE-REQUISITES

- Dockerfile in the project folder
- Azure Container Registry login credentials
- App Service for containers.

Below are the secrets and GitHub Actions that need to be added to GitHub workflow.

Secrets:

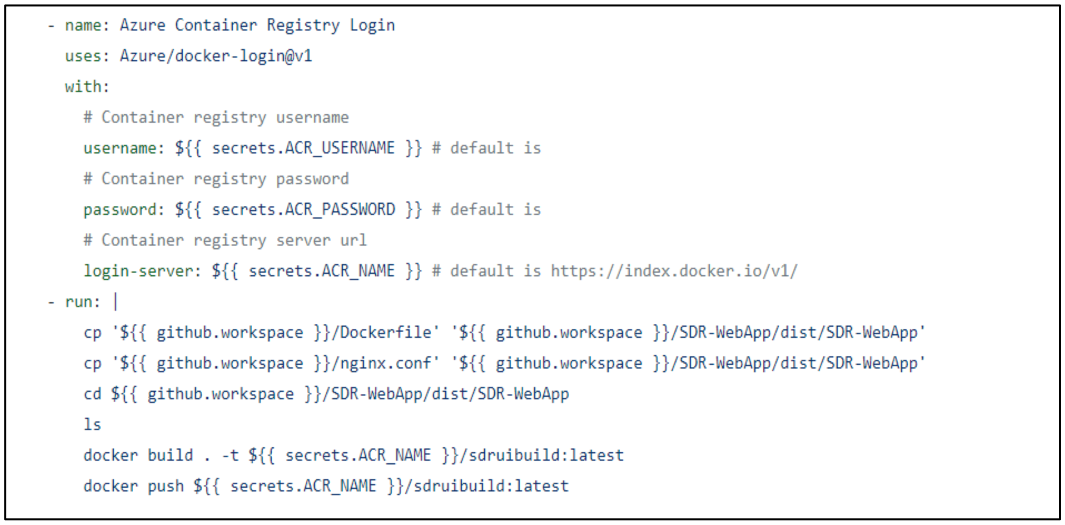
- ACR_NAME: Azure *Container Registry URL*
- ACR_USERNAME: *ACR Username*
- ACR_PASSWORD: *ACR Password*
- AZURE_WEBAPP_NAME: *UI App Service Name*

Workflow Actions:

- Build and push Docker image to ACR

The actions below will be used in deployment Job to build containers using an Azure Container Registry.

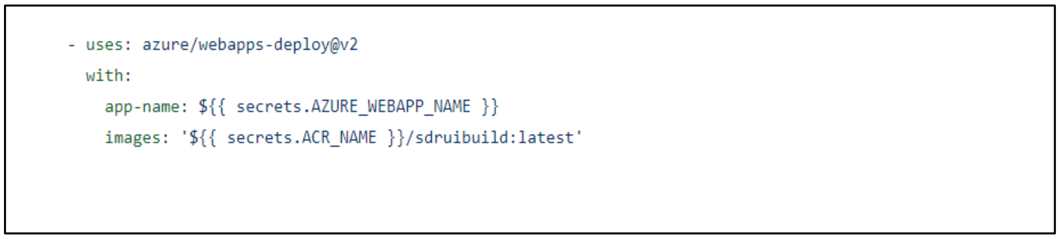
Figure 40 GitHub Action for Containerized Build



```
- name: Azure Container Registry Login
  uses: Azure/docker-login@v1
  with:
    # Container registry username
    username: ${{ secrets.ACR_USERNAME }} # default is
    # Container registry password
    password: ${{ secrets.ACR_PASSWORD }} # default is
    # Container registry server url
    login-server: ${{ secrets.ACR_NAME }} # default is https://index.docker.io/v1/
- run: |
    cp '${{ github.workspace }}/Dockerfile' '${{ github.workspace }}/SDR-WebApp/dist/SDR-WebApp'
    cp '${{ github.workspace }}/nginx.conf' '${{ github.workspace }}/SDR-WebApp/dist/SDR-WebApp'
    cd ${{ github.workspace }}/SDR-WebApp/dist/SDR-WebApp
    ls
    docker build . -t ${{ secrets.ACR_NAME }}/sdruibuild:latest
    docker push ${{ secrets.ACR_NAME }}/sdruibuild:latest
```

- Deploy to App Service

The below GitHub action will be used in deployment Job to deploy container to App Service.



```
- uses: azure/webapps-deploy@v2
  with:
    app-name: ${{ secrets.AZURE_WEBAPP_NAME }}
    images: '${{ secrets.ACR_NAME }}/sdruibuild:latest'
```

4.2.2 GitHub Actions to deploy container image to API App Service

The Azure Web Deploy action automates the workflow to deploy custom container image to app service.

PRE-REQUISITES

- Dockerfile in the project folder
- Azure Container Registry login credentials
- App Service for containers.

Below are the secrets and GitHub Actions that need to be added to GitHub workflow.

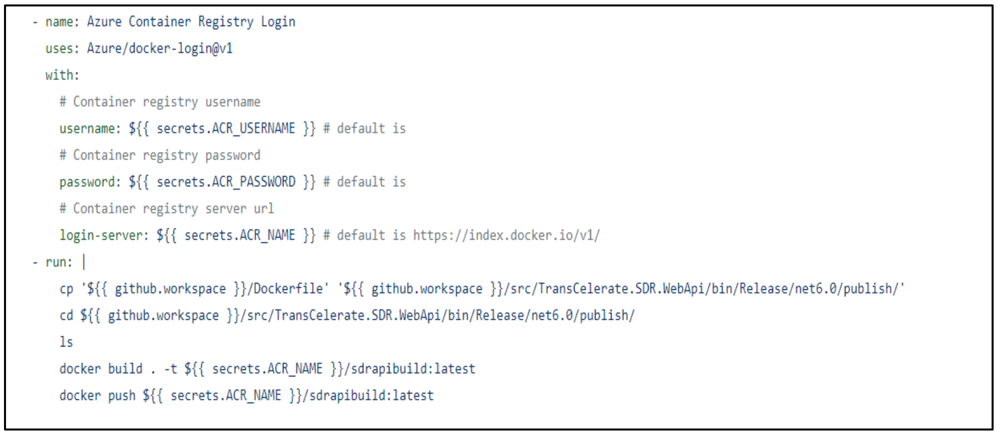
Secrets:

- ACR_NAME: Azure Container Registry URL
- ACR_USERNAME: ACR Username
- ACR_PASSWORD: ACR Password
- AZURE_WEBAPP_NAME: API Web App Name

Workflow Actions:

- Build and push Docker image to ACR.

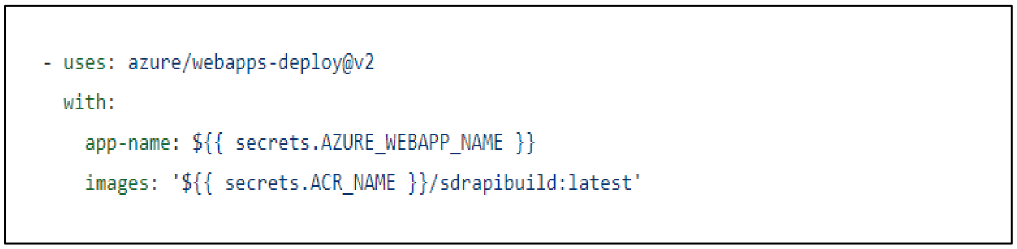
The actions below will be used in deployment Job to build containers using an Azure Container Registry.



```
- name: Azure Container Registry Login
  uses: Azure/docker-login@v1
  with:
    # Container registry username
    username: ${{ secrets.ACR_USERNAME }} # default is
    # Container registry password
    password: ${{ secrets.ACR_PASSWORD }} # default is
    # Container registry server url
    login-server: ${{ secrets.ACR_NAME }} # default is https://index.docker.io/v1/
- run: |
    cp '${{ github.workspace }}/Dockerfile' '${{ github.workspace }}/src/TransCelerate.SDR.WebApi/bin/Release/net6.0/publish/'
    cd ${{ github.workspace }}/src/TransCelerate.SDR.WebApi/bin/Release/net6.0/publish/
    ls
    docker build . -t ${{ secrets.ACR_NAME }}/sdrapibuild:latest
    docker push ${{ secrets.ACR_NAME }}/sdrapibuild:latest
```

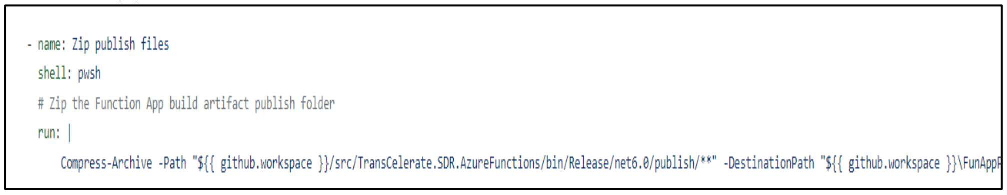
- Deploy to App Service

The below GitHub action will be used in deployment Job to deploy container to App Service.

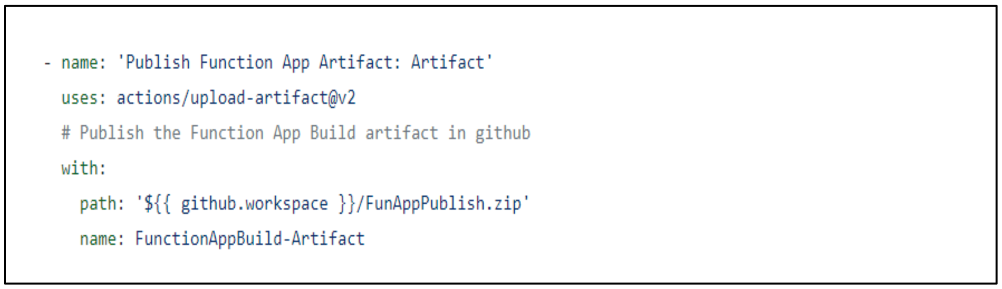


```
- uses: azure/webapps-deploy@v2
  with:
    app-name: ${{ secrets.AZURE_WEBAPP_NAME }}
    images: '${{ secrets.ACR_NAME }}/sdrapibuild:latest'
```

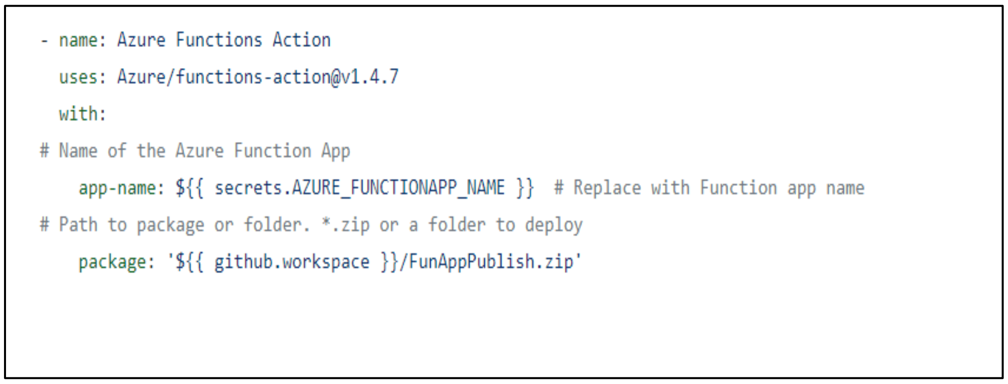
The below GitHub action will be used in build and deployment Job to Publish Function App folder.



```
- name: Zip publish files
  shell: pwsh
  # Zip the Function App build artifact publish folder
  run: |
      Compress-Archive -Path "${{ github.workspace }}/src/TransCelerate.SDR.AzureFunctions/bin/Release/net6.0/publish/**" -DestinationPath "${{ github.workspace }}\FunAppP
```



```
- name: 'Publish Function App Artifact: Artifact'
  uses: actions/upload-artifact@v2
  # Publish the Function App Build artifact in github
  with:
    path: '${{ github.workspace }}/FunAppPublish.zip'
    name: FunctionAppBuild-Artifact
```



```
- name: Azure Functions Action
  uses: Azure/functions-action@v1.4.7
  with:
# Name of the Azure Function App
    app-name: ${{ secrets.AZURE_FUNCTIONAPP_NAME }}  # Replace with Function app name
# Path to package or folder. *.zip or a folder to deploy
    package: '${{ github.workspace }}/FunAppPublish.zip'
```

4.2.3 GitHub Secrets used in Workflow

The Below mentioned additional GitHub Secrets need to be added in UI and API Repo Secrets.

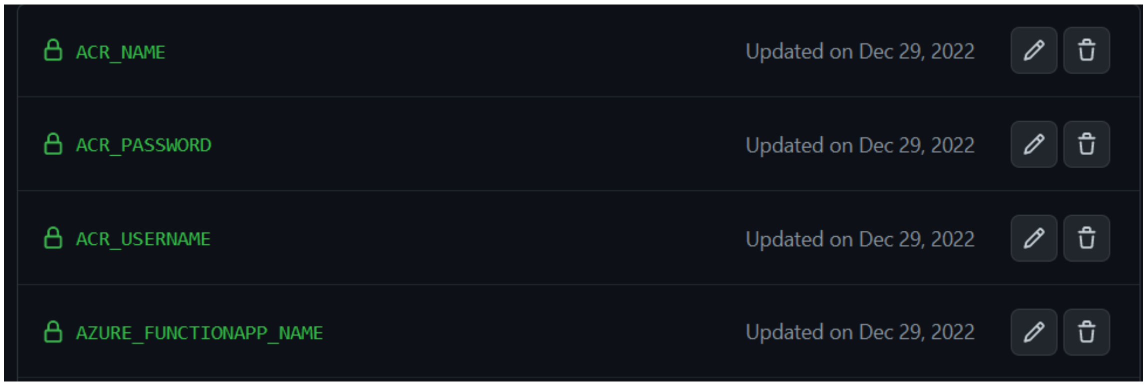
STEPS:

1. Login to azure portal, select resource group section.
2. Navigate to the deployed Azure Container Registry and copy the Login Server name from Overview blade. This will be the ACR_NAME for both ddf-sdr-ui & ddf-sdr-api repo secrets.
3. Navigate to the deployed Azure Container Registry and copy the Username & Password from Access Keys blade. This will be the

ACR_USERNAME & ACR_PASSWORD for both ddf-sdr-ui & ddf-sdr-api repo secrets.

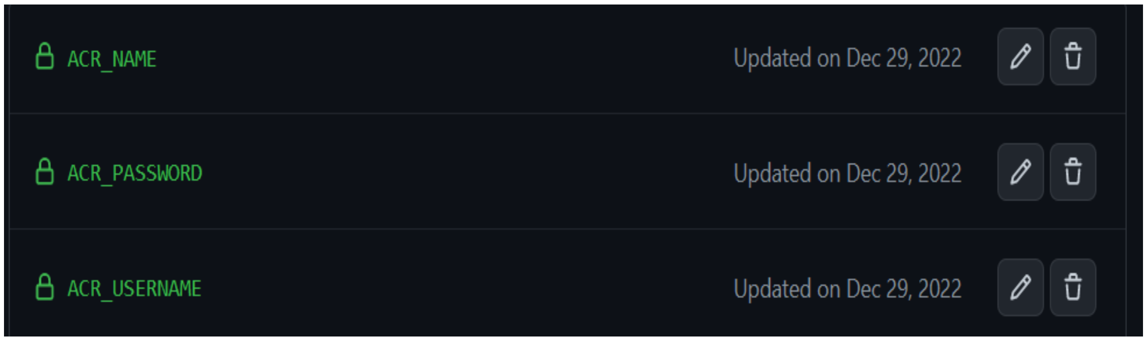
4. Navigate to the deployed Function App Service for SDR API and copy the name from Overview blade. This will be the AZURE_FUNCTIONAPP_NAME for ddf-sdr-api repo secrets.
5. Login to GitHub and navigate to ddf-sdr-api -> Settings-> Secrets -> Actions and add/replace values for ACR_NAME, ACR_USERNAME, ACR_PASSWORD, AZURE_FUNCTIONAPP_NAME by clicking on update.
6. Login to GitHub and navigate to ddf-sdr-ui -> Settings -> Secrets -> Actions and add/replace values for ACR_NAME, ACR_USERNAME, ACR_PASSWORD by clicking on Update.
7. Update the Deployment yml script with Azure Container Registry in both ddf-sdr-api & ddf-sdr-ui repos.

Figure 41 Container Registry GitHub Secrets for API



ACR_NAME	Updated on Dec 29, 2022
ACR_PASSWORD	Updated on Dec 29, 2022
ACR_USERNAME	Updated on Dec 29, 2022
AZURE_FUNCTIONAPP_NAME	Updated on Dec 29, 2022

Figure 42 Container Registry GitHub Secrets for UI App Service



ACR_NAME	Updated on Dec 29, 2022
ACR_PASSWORD	Updated on Dec 29, 2022
ACR_USERNAME	Updated on Dec 29, 2022