**Mendix Platform QA Setup**

Table Of Contents

[1. Overview 4](#_Toc171676880)

[2. Scope 4](#_Toc171676881)

[3. Azure Subscriptions 4](#_Toc171676882)

[4. Request Subscription Access 4](#_Toc171676883)

[5. Create Azure Virtual Network (VNet) 5](#_Toc171676884)

[6. Create Azure Kubernetes Service (AKS) Cluster 6](#_Toc171676885)

[6.1 Generate Command to Create AKS Cluster 7](#_Toc171676886)

[6.2 Connect to the AKS Cluster 8](#_Toc171676887)

[6.3 Configure Workload Identity on an Azure Kubernetes Service (AKS) cluster 8](#_Toc171676888)

[7. Create Azure Container Registry(ACR) 9](#_Toc171676889)

[7.1 Create TLS Secret 9](#_Toc171676890)

[7.2 Decrypt the private key before creating a secret by using the command 10](#_Toc171676891)

[7.3 Create TLS secret command 10](#_Toc171676892)

[7.4 Create TLS secret with full chain certificate for Azure application gateway 10](#_Toc171676893)

[8. Create an Ingress Controller 10](#_Toc171676894)

[8.1 Non-Supported Ingress Controllers 11](#_Toc171676895)

[8.2 Supported Ingress Controllers 11](#_Toc171676896)

[8.2.1 Import the images used by the Helm chart into your ACR 11](#_Toc171676897)

[8.2.2 Install NGINX Ingress Controller 12](#_Toc171676898)

[9 . Create Azure Application Gateway 13](#_Toc171676899)

[10 . Create a storage account 14](#_Toc171676900)

[Basics 14](#_Toc171676901)

[Advanced 14](#_Toc171676902)

[Networking 15](#_Toc171676903)

[Data protection 15](#_Toc171676904)

[Encryption 15](#_Toc171676905)

[11. Create SQL Managed Instance 15](#_Toc171676906)

[12. Mendix for Private Cloud Setup 16](#_Toc171676907)

[12.1 Creating a Cluster and Namespace 16](#_Toc171676908)

[12.2 Installing and Configuring the Mendix Operator 16](#_Toc171676909)

[12.3 Registry Configuration 17](#_Toc171676910)

[12.4 Running the mxpc Configuration Tool 19](#_Toc171676911)

[12.5 Verify Mendix Operator Running Status 22](#_Toc171676912)

[12.6 Mendix Portal 22](#_Toc171676913)

[12.6.1 Register in Mendix Portal 22](#_Toc171676914)

[12.6.2 Access Private Cloud 23](#_Toc171676915)

[12.6.3 Private Cluster Details 23](#_Toc171676916)

[13. Deploy a Mendix App to a Private Cloud Cluster 24](#_Toc171676917)

[13.1 Deploying a Mendix App with an Operator 24](#_Toc171676918)

[13.2 Verify the Mendix App Is Running 26](#_Toc171676919)

[14. Mendix Academy 26](#_Toc171676920)

[15. Mendix Support Ticket 27](#_Toc171676921)

[16. Mendix Installation Troubleshooting Steps 27](#_Toc171676922)

[16.1 General Check On Pods and Logs 27](#_Toc171676923)

[16.2 Networking Issue 27](#_Toc171676924)

[16.3 App Issue 27](#_Toc171676925)

[16.4 Deployment Issue 27](#_Toc171676926)

[16.5 Port Forwarding To Test Mendix App in localhost 27](#_Toc171676927)

# Overview

Creating a Private AKS Cluster to manage the deployment of Mendix apps in QA environment.

# Scope

The scope of work for the cloud team is to setup the QA environment in Honeywell Azure Cloud compliant with Honeywell security standards.

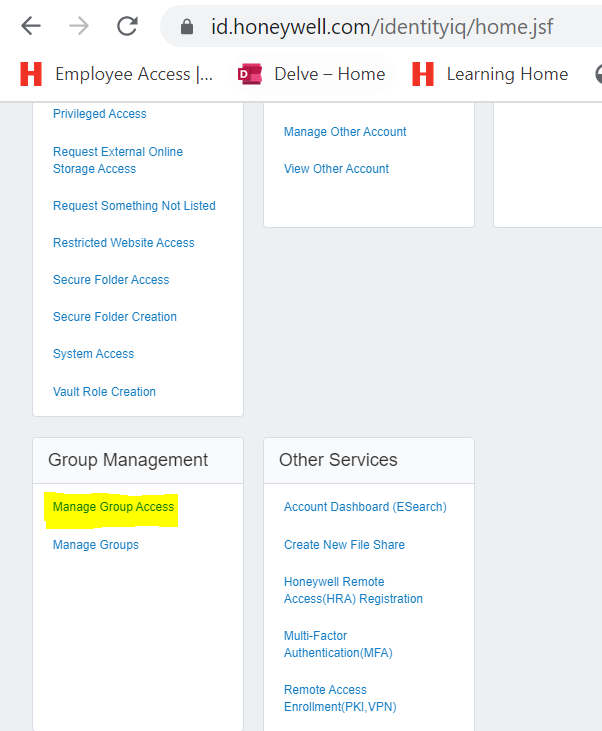
# Azure Subscriptions

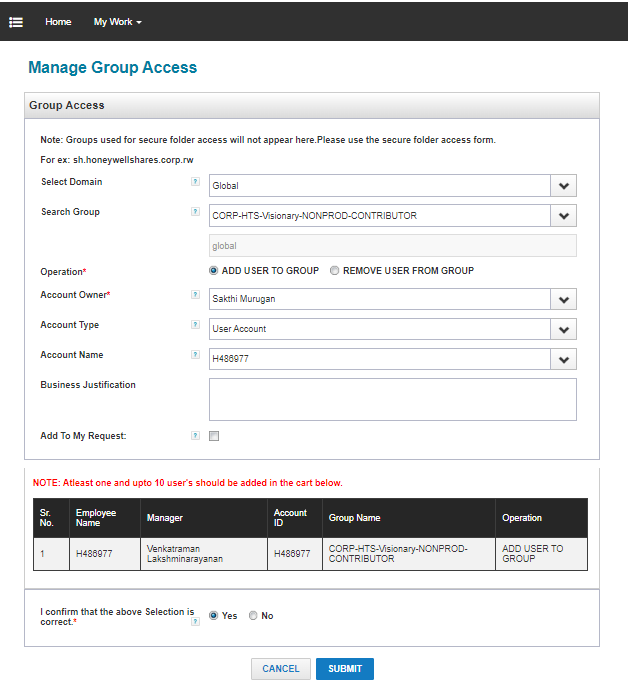
The azure subscriptions for visionary platform application are listed below

* NonProd - CORP-ENIT-MENDIX-NONPROD
* Prod - CORP-ENIT-MENDIX-PROD

# Request Subscription Access

To gain subscription access, user must raise a request in id.honeywell.com, under Group management select “Manage Group Access” and raise a request to add to the group CL.AZRCOM.CORP-IMA-MENDIX.NONPROD-CONTRIBUTOR as shown below

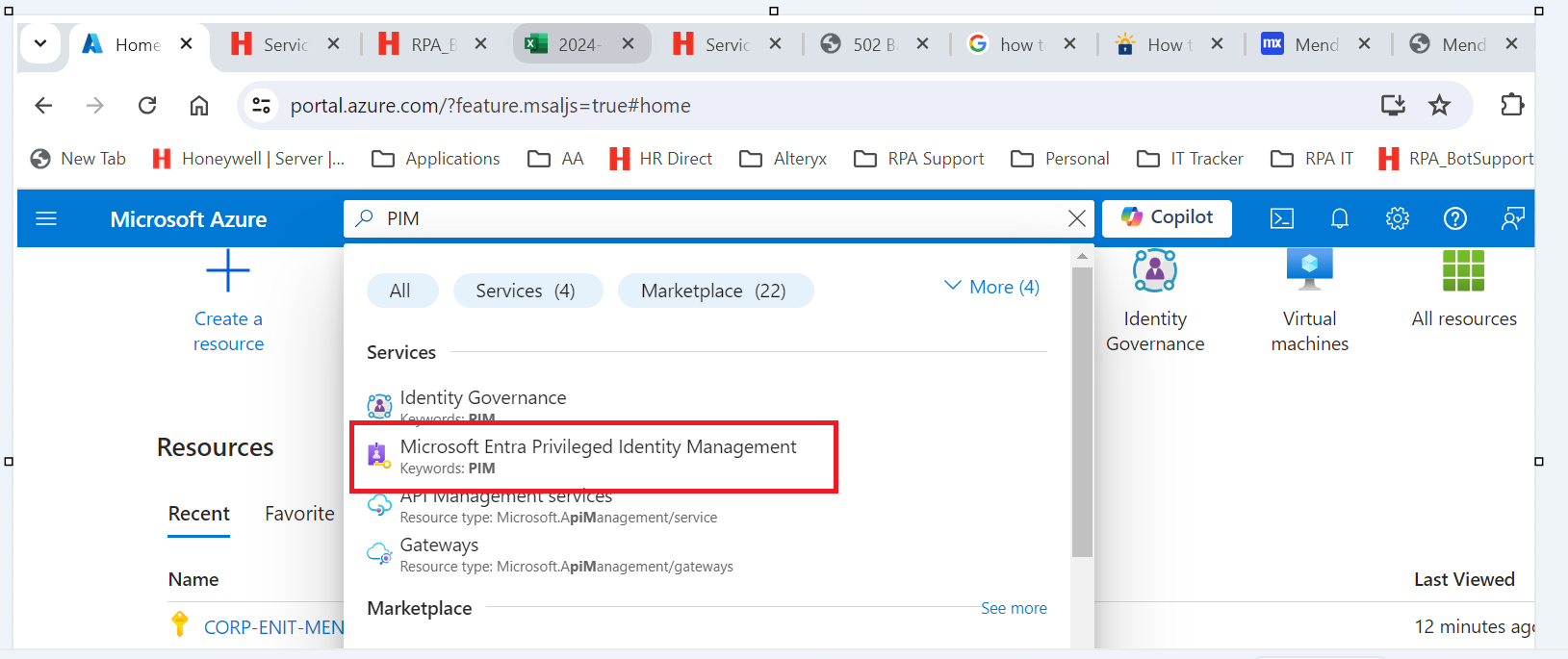




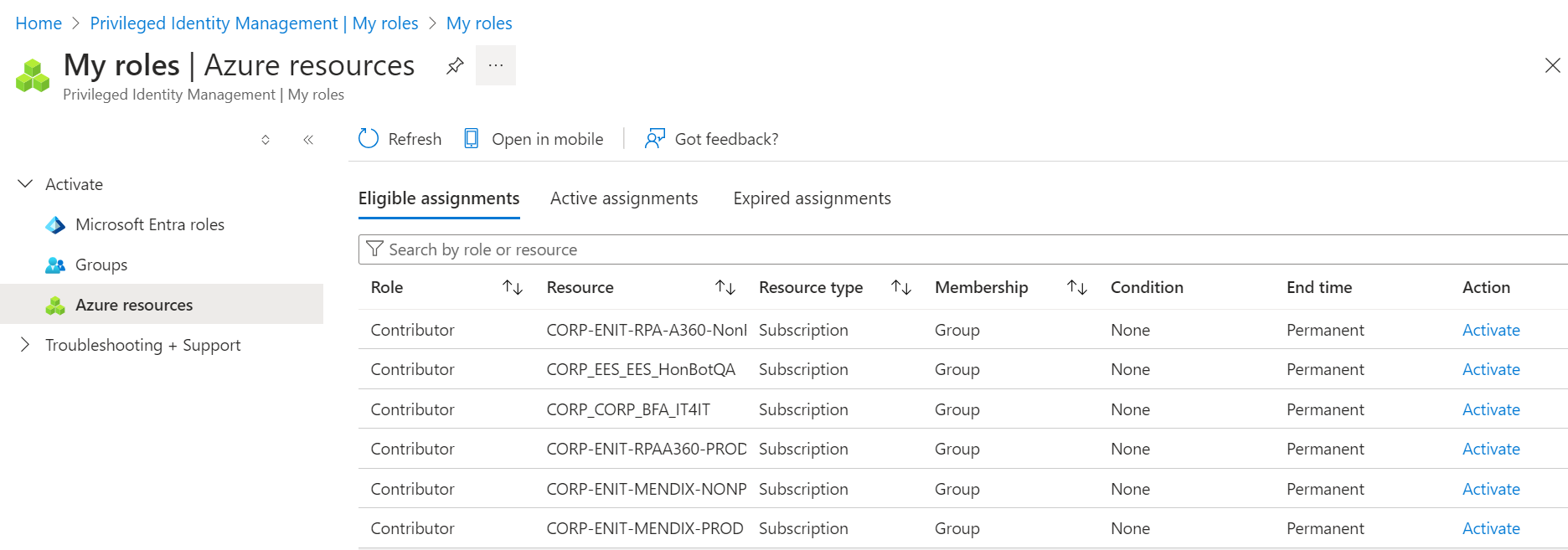
The request approval will go to the app team and once approved the user will be able to see the azure subscription (as shown below in SS) in the Azure portal (<https://portal.azure.com>).

Activate Subscription using PIM(Privileged Identity Management) as highlighted.

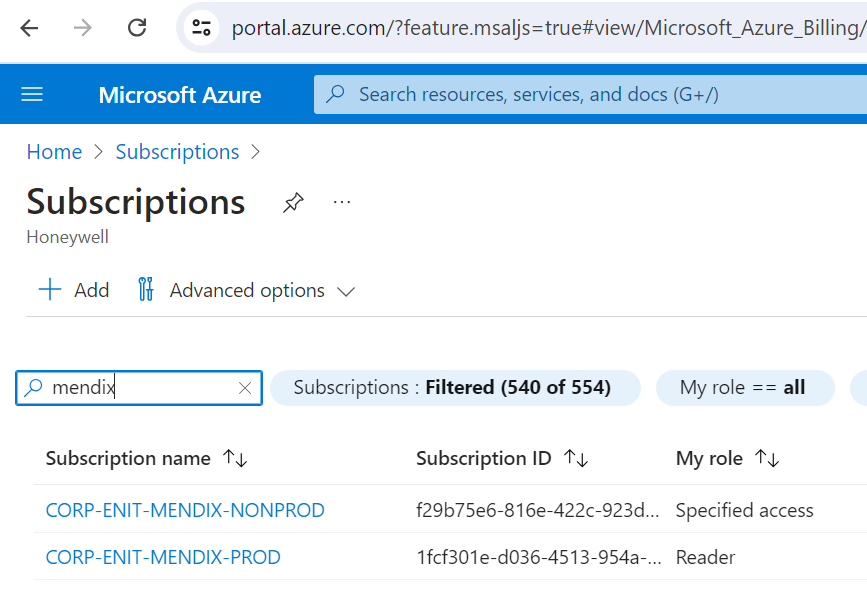
Go to roles>Azure Resources> <Select the required subscription>>Click on Activate.



Activate Subscription: -



Validate subscription



# Create Azure Virtual Network (VNet)

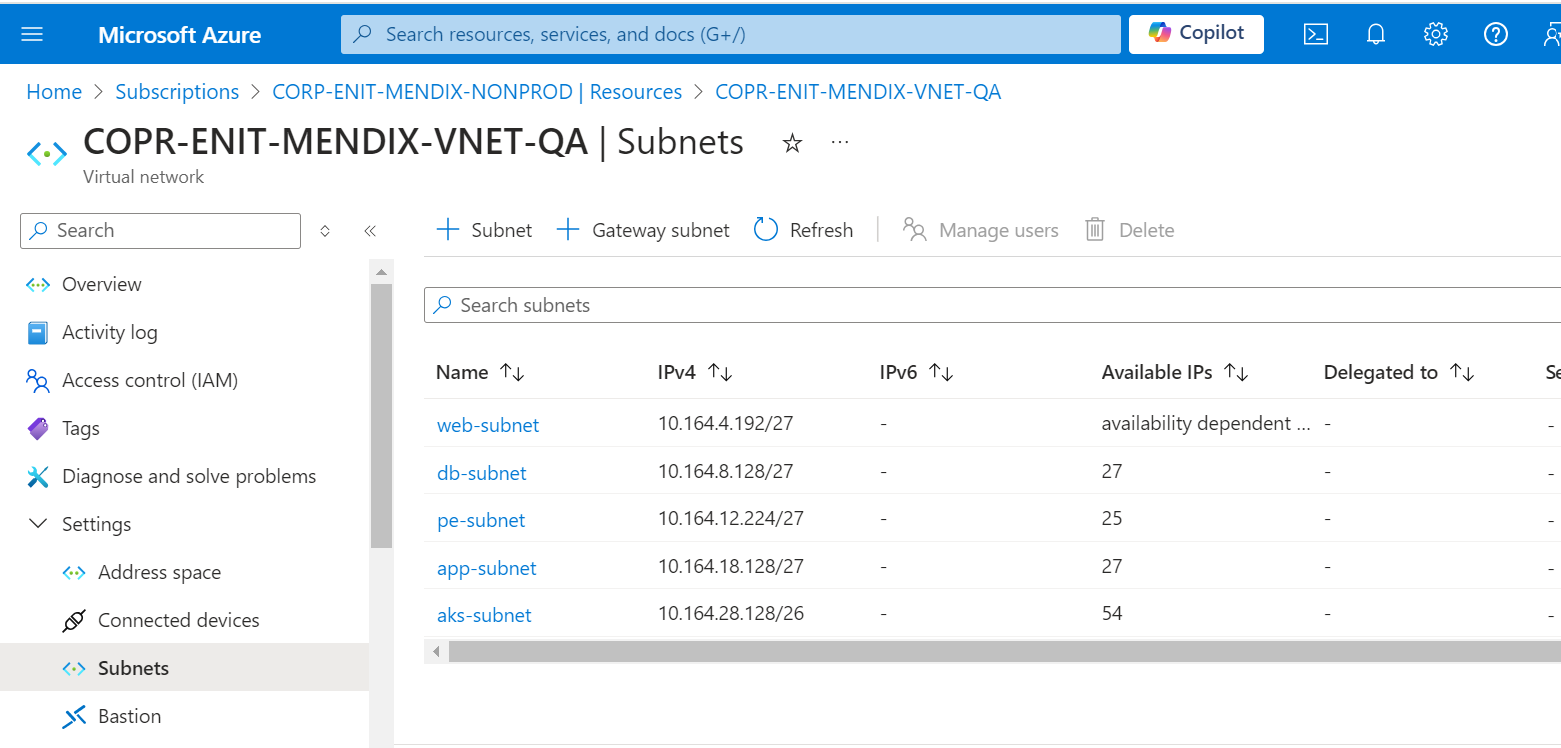
Raise a request for a new vnet in ITDirect  <https://honeywell.service-now.com/itdirect?id=sc_cat_item&sys_id=ecb86f271bec7010f5addc26bc4bcb6f> see **RITM9526401 for ref.**

Azure Commercial: Virtual Network Request with the below details

* Requested For – Name of the requester
* Subscription Name - CORP-ENIT-MENDIX-NONPROD
* Environment – NON-PROD
* Region – East US
* New vNET Name - COPR-ENIT-MENDIX-VNET-DEV
* Number of Subnets Required – 5

|  |  |  |
| --- | --- | --- |
| **S.No** | **New Subnet Name to be created** | **Select the required subnet size** |
| 1 | web-subnet | 27 |
| 2 | db-subnet | 27 |
| 3 | pe-subnet | 27 |
| 4 | app-subnet | 27 |
| 5 | aks-subnet | 59 |

The request will go to the Cloud Infrastructure team (DL-CORPEHCAzureAdmins@Honeywell.com) and once approved the new vNet will be created and details will look like the figure shown below



Reference Link: <https://confluence.honeywell.com/display/EHC/Azure%3A+Subscription+Walkthrough>

# Create Azure Kubernetes Service (AKS) Cluster

Azure Kubernetes Service (AKS) offers the quickest way to start developing and deploying cloud native apps, with built-in code-to-cloud pipelines and guardrails. Get unified management and governance for on-premises, edge, and multi cloud Kubernetes clusters. Interoperate with Azure security, identity, cost management, and migration services.

To create Azure Kubernetes Service (AKS) cluster, please follow the deployment instructions provided by the Cloud Infrastructure available

here:  [https://confluence.honeywell.com/display/EHC/Azure+2.0+AKS+Reference+Architecture](https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fconfluence.honeywell.com%2Fdisplay%2FEHC%2FAzure%2B2.0%2BAKS%2BReference%2BArchitecture&data=05%7C02%7CSakthi.Murugan%40Honeywell.com%7C80aed6be93eb4d59778108dc70f2a978%7C96ece5269c7d48b08daf8b93c90a5d18%7C0%7C0%7C638509434737765419%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C0%7C%7C%7C&sdata=%2B3o6Qdlbp0oQXiMQU%2F5PHjGIQ1kkTedz9DAT7Z0iGj4%3D&reserved=0)

Steps to create the AKS cluster.

* Refer section Deployment Steps in the above-mentioned [link](https://confluence.honeywell.com/display/EHC/Azure+2.0+AKS+Reference+Architecture) and complete all the steps listed below
* Set up the Azure CLI
* Create Resource Group
* Create a Managed Identity
* Get the Azure Resource ID of the Azure 2.0 Subnet
  + Raise IT Direct ticket for role assignment to subscription level & Manage Identity level

Role name: HonAKSCustomRole

<https://honeywell.service-now.com/itdirect?id=sc_cat_item&sys_id=e92d0ac61bb87490f5addc26bc4bcbcf>

reference ticket - RITM9533174 & RITM9533206

* Create Cluster

## Generate Command to Create AKS Cluster

az aks create -n "aksclustermendixq" -g " rg-k8s-reference-QA" \

--subscription "f29b75e6-816e-422c-923d-2f13b09352e1" \

--load-balancer-sku standard \

--enable-private-cluster \

--enable-managed-identity \

--assign-identity "/subscriptions/f29b75e6-816e-422c-923d-2f13b09352e1/resourcegroups/rg-k8s-reference-QA/providers/Microsoft.ManagedIdentity/userAssignedIdentities/mi-enit-mendix-np-qa" \

--network-plugin "azure" \

--network-plugin-mode "overlay" \

--vnet-subnet-id "/subscriptions/f29b75e6-816e-422c-923d-2f13b09352e1/resourceGroups/COPR-ENIT-MENDIX-VNET-QA/providers/Microsoft.Network/virtualNetworks/COPR-ENIT-MENDIX-VNET-QA/subnets/aks-subnet" \

--service-cidr 172.26.0.0/24 \

--dns-service-ip 172.26.0.10 \

--pod-cidr 172.27.0.0/16 \

--outbound-type userDefinedRouting \

--disable-public-fqdn \

--network-policy "cilium" \

--network-dataplane "cilium" \

--generate-ssh-keys \

--enable-aad \

--enable-azure-rbac \

--aad-admin-group-object-ids "7cd362fb-0587-451e-8cce-77bfd2626943" \

--disable-local-accounts \

--enable-addons azure-policy \

--enable-addons azure-keyvault-secrets-provider \

--windows-admin-username "aksadminmendixq" \

--windows-admin-password "Mendi%QaAdm1Ndas#" \

--vm-set-type VirtualMachineScaleSets

Note: **Pass your own values for the parameters before execution**.

Resources used to generate the command are listed below

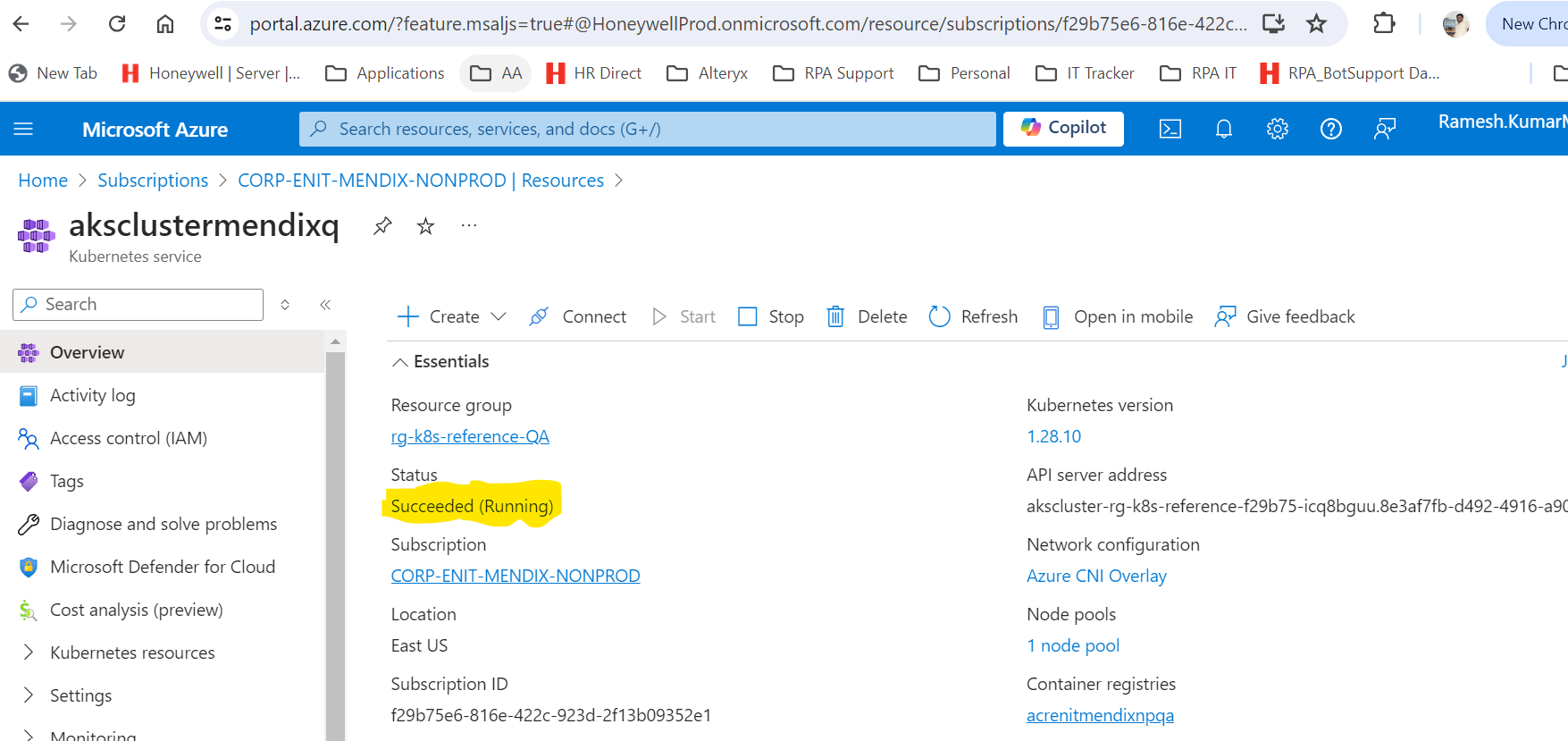
* Subscription - CORP-ENIT-MENDIX-NONPROD
* Managed Identity - mi-enit-mendix-np-qa
* Subnet – aks-subnet
* Admin AD Group Name - App.AksAdmin.MendixDev(7cd362fb-0587-451e-8cce-77bfd2626943)
* Gather DNS Info of Kubernetes API Server (FQDN/IP)

**ATTENTION: AKS Cluster deployment status will not become available until the Register Private DNS step below is complete.**

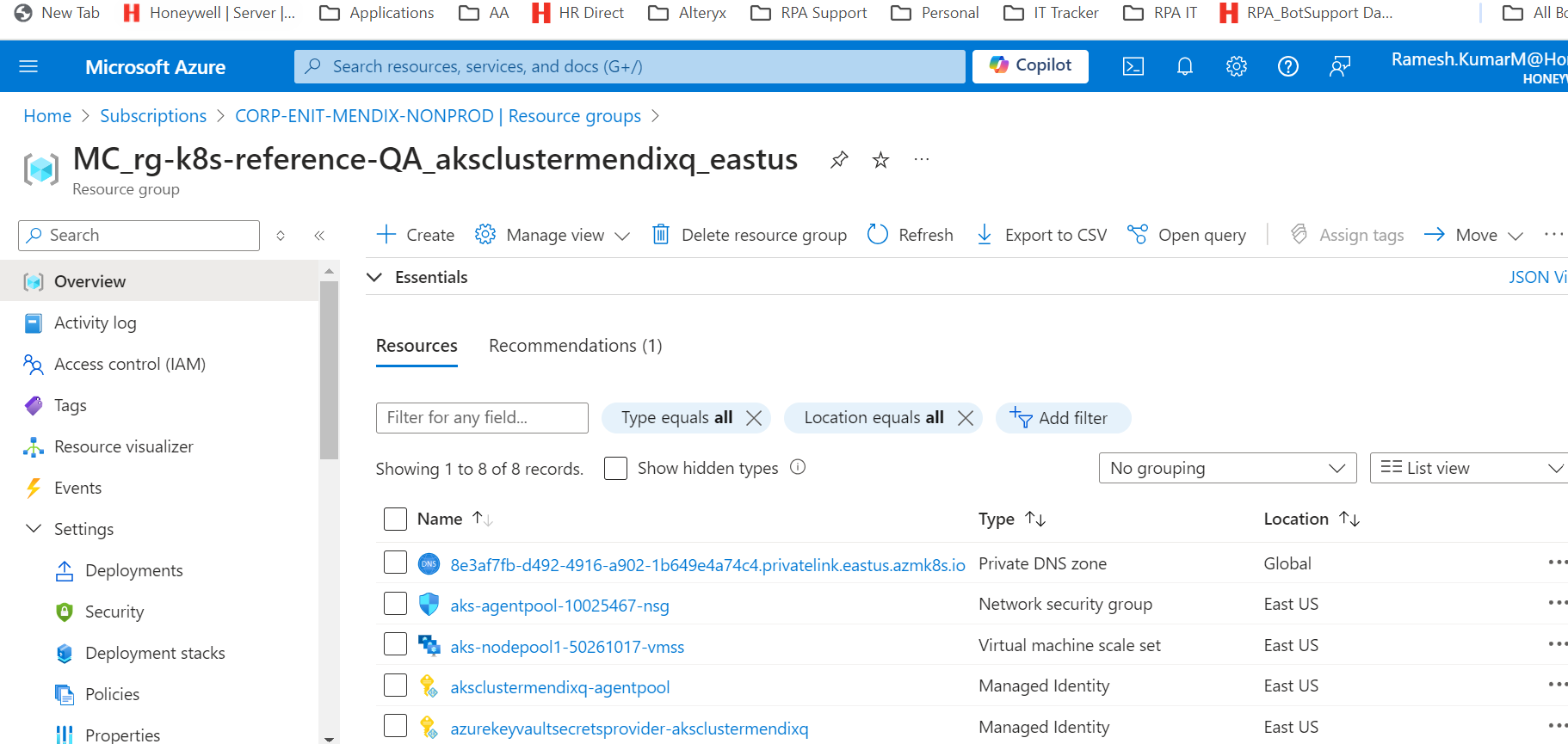
* Register Private DNS
  + Raise IT Direct Ticket in below link – Reference ticket - RITM9553949

<https://honeywell.service-now.com/itdirect?id=sc_cat_item&sys_id=5e4a7500dbbd430092773220ad96190a>

Make sure the AKS cluster status as “Succeeded (Running)”



AKS Cluster creates a new resource group(MC\_rg-k8s-reference-QA\_aksclustermendixq\_eastus) which will have all the dependent resources.



## Connect to the AKS Cluster

Follow steps mentioned in the [link](https://confluence.honeywell.com/display/EHC/Azure+2.0+AKS+Reference+Architecture) to connect to the AKS cluster. Refer section “Connect to the AKS Cluster”.

Install kubectl locally using the *[az aks install-cli](https://learn.microsoft.com/en-us/cli/azure/aks" \l "az-aks-install-cli)* command:

## Configure Workload Identity on an Azure Kubernetes Service (AKS) cluster

Follow steps mentioned in the link <https://learn.microsoft.com/en-us/azure/aks/workload-identity-deploy-cluster>

Skip steps

* Create a resource group
* Create an AKS cluster and
* Disable workload identity

Commands used

az aks update --resource-group " rg-k8s-reference-QA" --name "aksclustermendixq" --enable-oidc-issuer --enable-workload-identity

set AKS\_OIDC\_ISSUER="$(az aks show --name "aksclustermendixq" --resource-group " rg-k8s-reference-QA" --query "oidcIssuerProfile.issuerUrl" --output tsv)"

az identity create --name "mxUserIdentity" --resource-group " rg-k8s-reference-QA" --location "eastus" --subscription "CORP-ENIT-MENDIX-NONPROD"

az identity show --resource-group " rg-k8s-reference-QA" --name "mxUserIdentity" --query 'clientId' --output tsv

Create workload-identity-sa.yml file 

kubectl apply -f workload-identity-sa.yml

# Create Azure Container Registry(ACR)

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.

Follow steps mentioned in the [link](https://confluence.honeywell.com/display/EHC/Azure+2.0+AKS+Reference+Architecture) to Create a new Azure Container Registry (ACR). Refer section “Integrate Azure Container Registry (ACR) “

Command used to create ACR and attach to the AKS cluster are listed below

az acr create -n acrenitmendixnpqa -g rg-k8s-reference-QA --sku premium

az aks update -n aksclustermendixq -g rg-k8s-reference-QA --attach-acr acrenitmendixnpqa

Create IT Direct ticket with Cloud Team for attaching ACR to AKS – Ticket Ref. RITM9574316

(Below step will be performed by Cloud Admin Team Only) - <https://honeywell.service-now.com/itdirect?id=sc_cat_item&sys_id=e92d0ac61bb87490f5addc26bc4bcbcf>

az aks update --name aksclustermendixq --resource-group rg-k8s-reference-QA --attach-acr acrenitmendixnpqa

### 7.1 Create TLS Secret

Kubernetes provides a built-in Secret type kubernetes.io/tls for storing a certificate and its associated key that are typically used for TLS.

One common use for TLS secrets is to configure encryption in transit for an Ingress.

Decrypt the private key

### 7.2 Decrypt the private key before creating a secret by using the command

Step: Ensure OpenSSL is installed on your system - [Download Openssl](https://openssl-library.org/source/)

winget install -e --id ShiningLight.OpenSSL

Navigate to openssl exe path (C:\OpenSSL\bin)

Download the SSL certificates from Venify portal - [Link](https://requestcert.honeywell.com/aperture/tls-protect/dashboard/certificate-dashboard)

Open the Openssl.exe

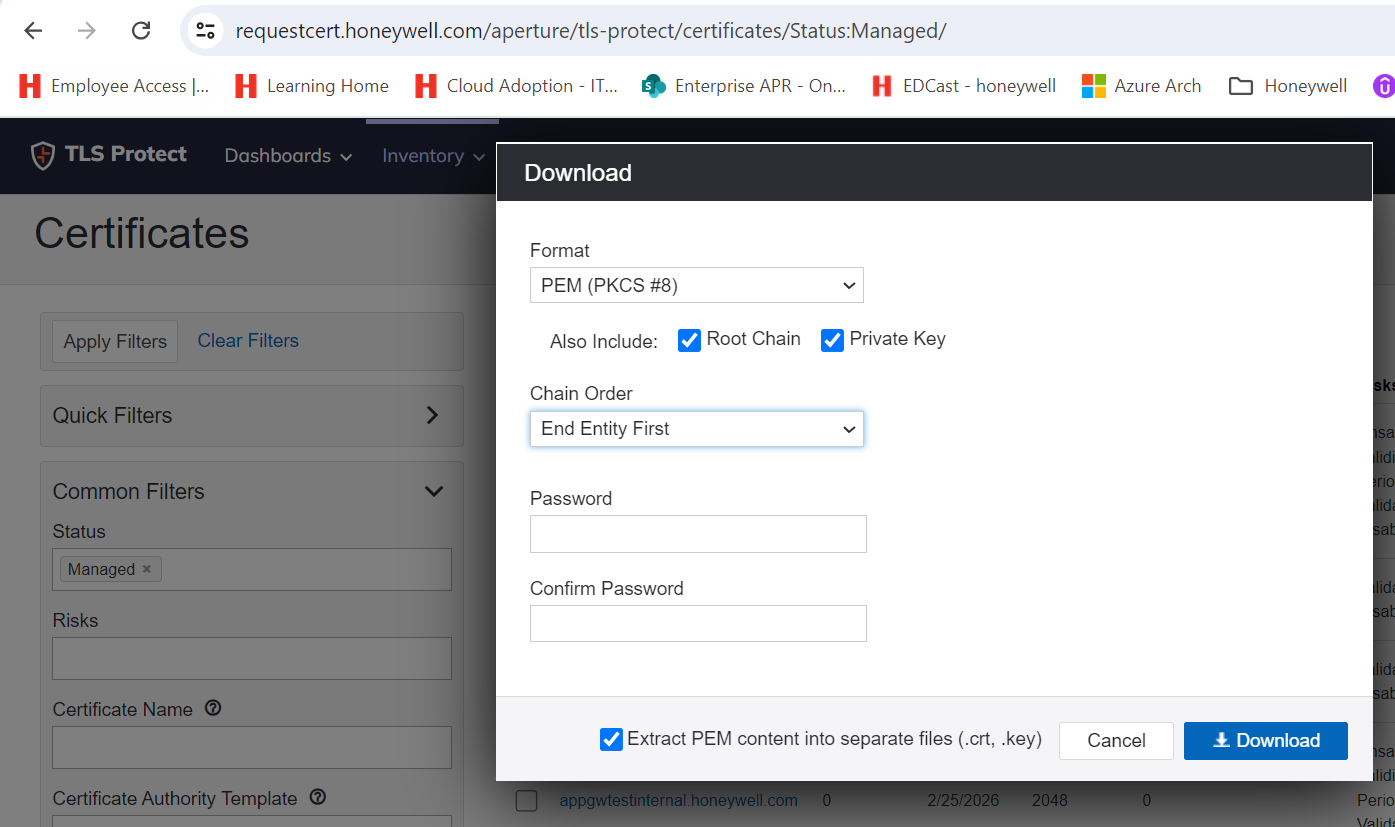
openssl rsa -in QMendix.honeywell.com.key -out newQAMendixPrivateKey.key

### 7.3 Create TLS secret command

kubectl create secret tls mendixqa-tls-secret --cert=QMendix.honeywell.com.crt --key= newQAMendixPrivateKey.key

### 7.4 Create TLS secret with full chain certificate for Azure application gateway

* Download certificate with option PEM (PKCS #8) as shown below



* Once the certificate .zip file is downloaded. Make sure we manually create a full chain certificate in the order
* Server Certificate
* Intermediate CA Certificate
* Root CA Certificate

Sample full chain certificate is attached here



* Run the command given below and make sure you use the fullchain-cert.crt file

kubectl create secret tls upsell-mendixqa-tls-secret --cert= **FullchainQMendix.honeywell.com.crt** --key= Qmendix.honeywell.com-decrypt.key

# Create an Ingress Controller

The networking type used in our AKS cluster is “Azure CNI Overlay”.

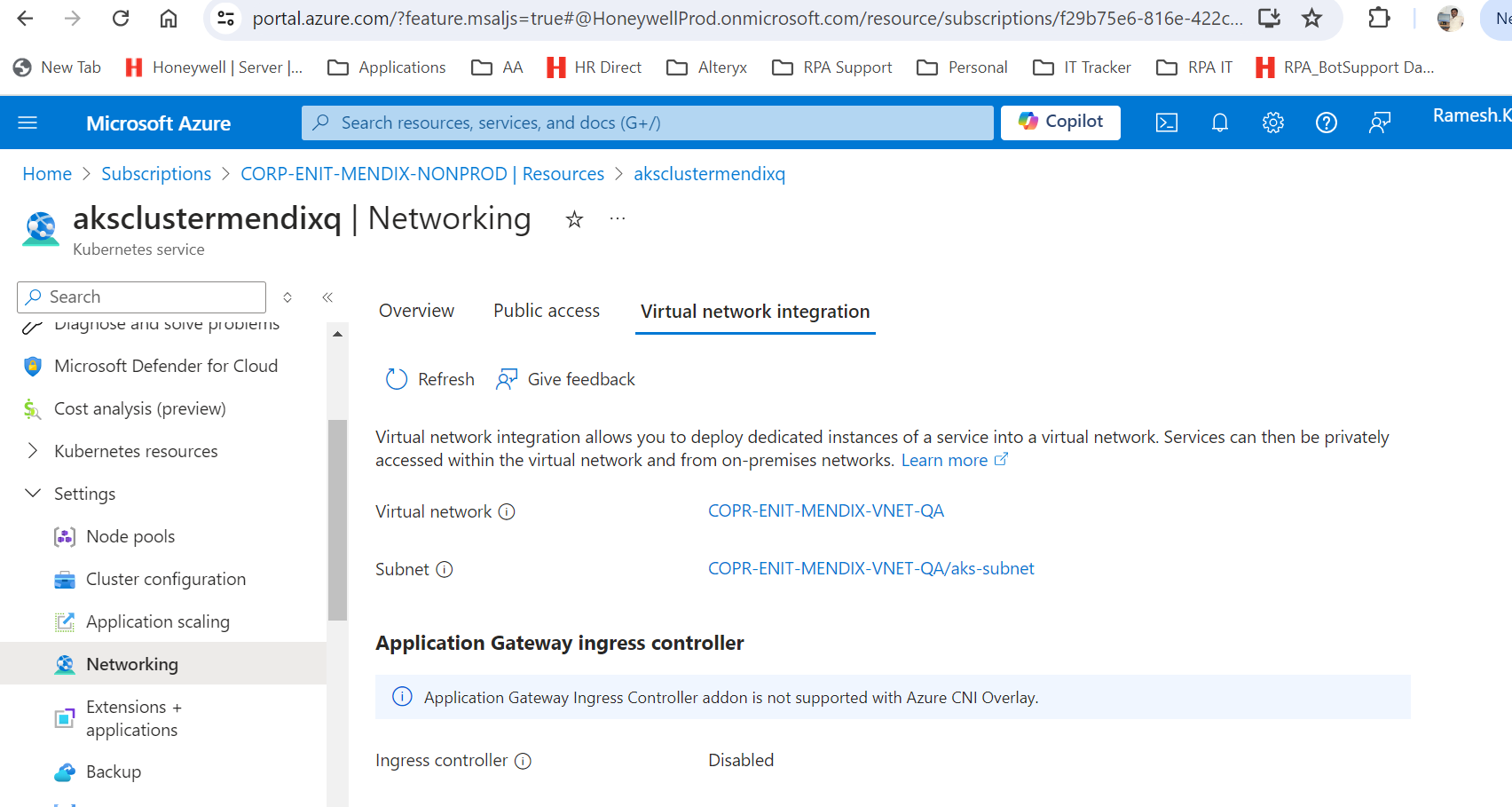
With Azure CNI Overlay, the cluster nodes are deployed into an Azure Virtual Network (VNet) subnet.

Reference Link - <https://learn.microsoft.com/en-us/azure/aks/azure-cni-overlay?tabs=kubectl#overview-of-overlay-networking>

## Non-Supported Ingress Controllers

There are two types of Ingress controller that are not supported with Azure CNI Overlay which are listed below

* Application Gateway for Containers
* Application Gateway Ingress Controller



## Supported Ingress Controllers

The supported Ingress controller is NGINX.

Reference Link - <https://learn.microsoft.com/en-us/azure/aks/ingress-basic?tabs=azure-cli#create-an-ingress-controller>

### Import the images used by the Helm chart into your ACR

To control image versions, you'll want to import them into your own Azure Container Registry. The NGINX ingress controller Helm chart relies on three container images. Use az acr import to import those images into your ACR.

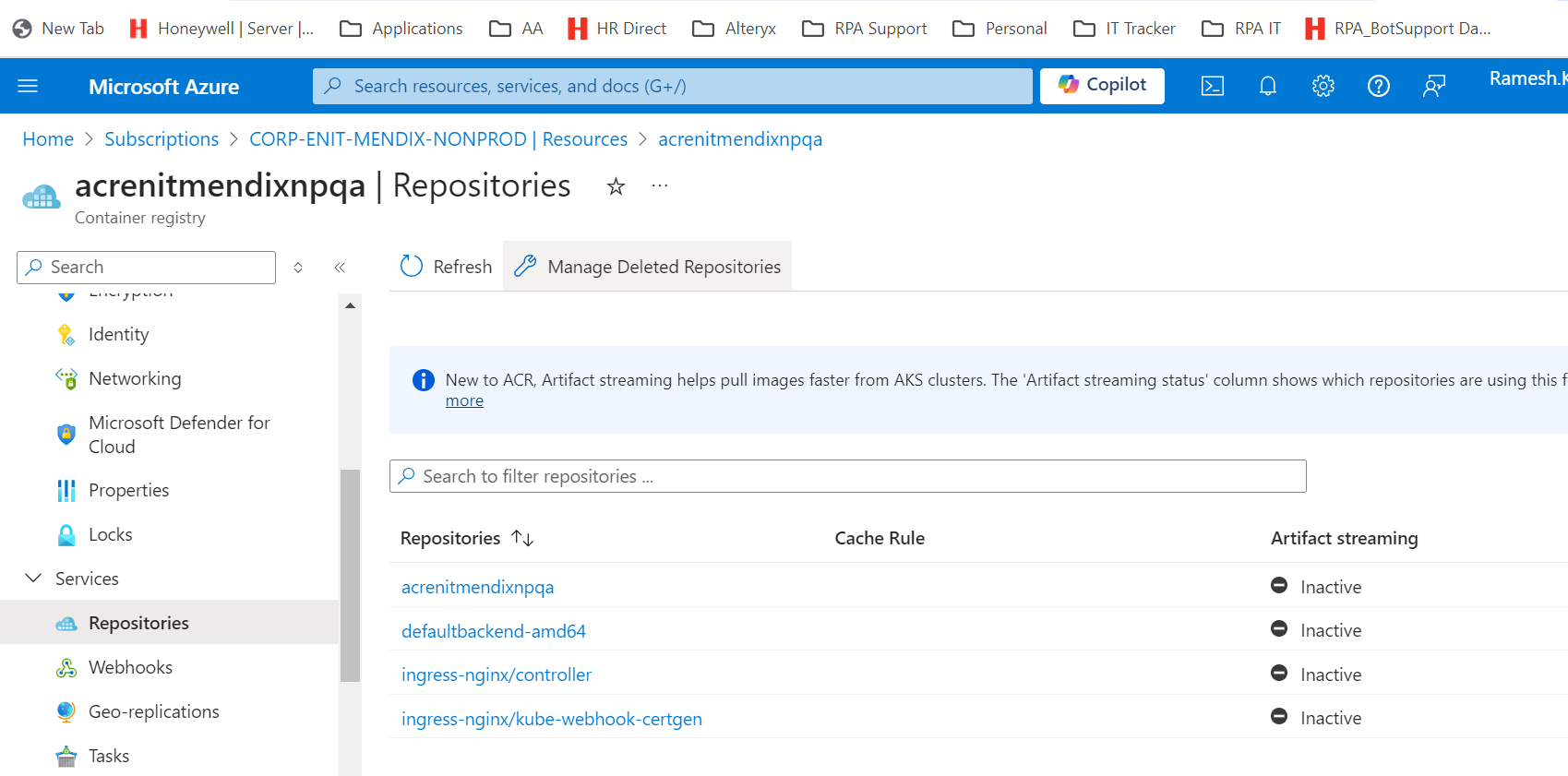
Reference Link - <https://learn.microsoft.com/en-us/azure/aks/ingress-basic?tabs=azure-cli#import-the-images-used-by-the-helm-chart-into-your-acr>

az acr import --name "acrenitmendixnpqa" --source registry.k8s.io/ingress-nginx/controller:v1.8.1 --image ingress-nginx/controller:v1.8.1

az acr import --name " acrenitmendixnpqa " --source registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20230407 --image ingress-nginx/kube-webhook-certgen:v20230407

az acr import --name " acrenitmendixnpqa " --source registry.k8s.io/defaultbackend-amd64:1.5 --image defaultbackend-amd64:1.5

Verify the images are pulled into the ACR



### Install NGINX Ingress Controller

Reference Link <https://confluence.honeywell.com/display/EHC/Azure+2.0+AKS+Reference+Architecture#Azure2.0AKSReferenceArchitecture-IntegrateAzureContainerRegistry(ACR)> and Section – Cluster Ingress

Install NGINX Ingress Controller using the below commands

Install helm using below command

winget install Helm.Helm

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

helm repo update

#Deploy the internal ingress controller and bind the certificate to it-ensure to run the below command in one line if using CMD.

helm install ingress-nginx ingress-nginx/ingress-nginx --debug --version 4.7.1 \

--namespace "ingress-basic" \

--set controller.replicaCount=2 \

--set controller.nodeSelector."kubernetes\.io/os"=linux \

--set controller.image.registry=acrenitmendixnpqa.azurecr.io \

--set controller.image.image=ingress-nginx/controller \

--set controller.image.tag=v1.8.1 \

--set controller.image.digest="" \

--set controller.admissionWebhooks.patch.nodeSelector."kubernetes\.io/os"=linux \

--set controller.service.loadBalancerIP=10.164.47.190 \

--set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-internal"=true \

--set controller.service.annotations."service\.beta\.kubernetes\.io/azure-load-balancer-health-probe-request-path"=/healthz \

--set controller.admissionWebhooks.patch.image.registry=acrenitmendixnpqa.azurecr.io \

--set controller.admissionWebhooks.patch.image.image=ingress-nginx/kube-webhook-certgen \

--set controller.admissionWebhooks.patch.image.tag=v20230407 \

--set controller.admissionWebhooks.patch.image.digest="" \

--set defaultBackend.nodeSelector."kubernetes\.io/os"=linux \

--set defaultBackend.image.registry= acrenitmendixnpqa.azurecr.io \

--set defaultBackend.image.image=defaultbackend-amd64 \

--set defaultBackend.image.tag=1.5 \

--set defaultBackend.image.digest=""

Output

----------

Installed successfully

kubectl --namespace ingress-basic get services -o wide -w ingress-nginx-controller

NOTES:

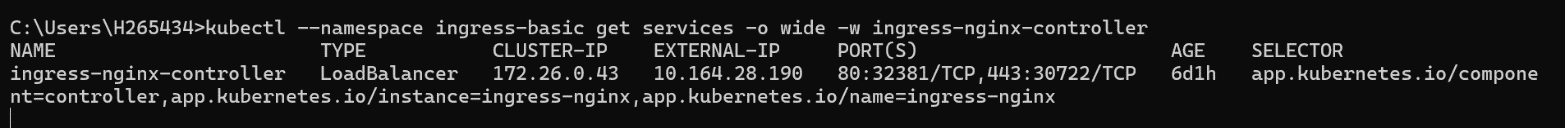
The ingress-nginx controller has been installed.

It may take a few minutes for the LoadBalancer IP to be available.

You can watch the status by running 'kubectl --namespace ingress-basic get services -o wide -w ingress-nginx-controller'

Verify NGINX ingress controller deployed properly using the below command

kubectl --namespace ingress-basic get services -o wide -w ingress-nginx-controller



# 9 . Create Azure Application Gateway

To create an Azure application gateway and map the Kubernetes internal load balancer as a backend, a request must be raised to the Cloud team from the link given below

*For****Support****on****Cloud Adoption & Enablement****, use the below links:*

***IT Direct Requests****->*[*Others - Cloud Enablement Request*](https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhoneywell.service-now.com%2Fitdirect%3Fid%3Dsc_cat_item%26sys_id%3Dd0d053bd1bda5150aaf2411abc4bcb2a&data=05%7C01%7CSakthi.Murugan%40Honeywell.com%7C02654b01faa5439ea2d608db0b2d7c0e%7C96ece5269c7d48b08daf8b93c90a5d18%7C0%7C0%7C638116062205174460%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=h0h68Ds%2BlN2jo8eCSYawshhlraQZmAnDSWfOcqa3QIo%3D&reserved=0)

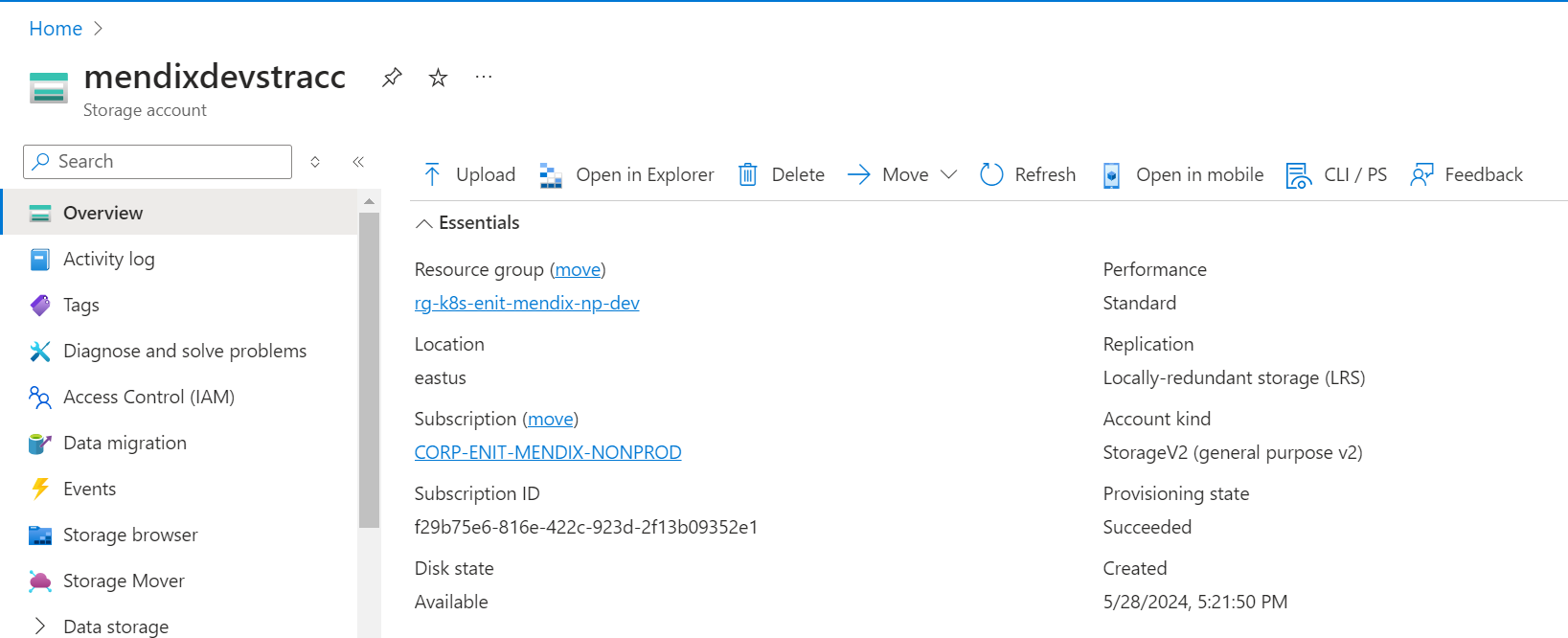
 See [**RITM9639663**](https://honeywell.service-now.com/itdirect?id=hsp_order&table=sc_req_item&sys_id=84cf14c747238e940710e29c416d437f) for reference

Application Gateway Resource Name (QA) – **mendixqaappgw**

# 10 . Create a storage account

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables.

Create a storage account which will be used to configure the Storage Plan for the Mendix setup.



Fill out the details as shown below and create

### Basics

Subscription CORP-ENIT-MENDIX-NONPROD

Resource Group rg-k8s-reference-QA

Location East US

Storage account name mendixqastracc

Deployment model Resource manager

Performance Standard

Replication Locally-redundant storage (LRS)

### Advanced

Secure transfer Enabled

Allow storage account key access Enabled

Allow cross-tenant replication Enabled

Default to Azure Active Directory authorization in the Azure portal Disabled

Blob public access Disabled

Minimum TLS version 1.2

Permitted scope for copy operations (preview) From any storage account

Enable hierarchical namespace Disabled

Enable network file system v3 Disabled

Access tier Hot

Enable SFTP (preview) Disabled

Large file shares Disabled

### Networking

Network connectivity Public endpoint (disabled)

Virtual network subscription CORP-ENIT-MENDIX-NONPROD

Virtual network resource group COPR-ENIT-MENDIX-VNET-QA

Virtual network COPR-ENIT-MENDIX-VNET-QA

Subnet pe-subnet

Default routing tier Microsoft network routing

Endpoint type Standard

### Data protection

Point-in-time restore Disabled

Blob soft delete Enabled

Blob retainment period in days 7

Container soft delete Enabled

Container retainment period in days 7

File share soft delete Enabled

File share retainment period in days 7

Versioning Disabled

Blob change feed Disabled

Version-level immutability support Disabled

### Encryption

Encryption type Microsoft-managed keys (MMK)

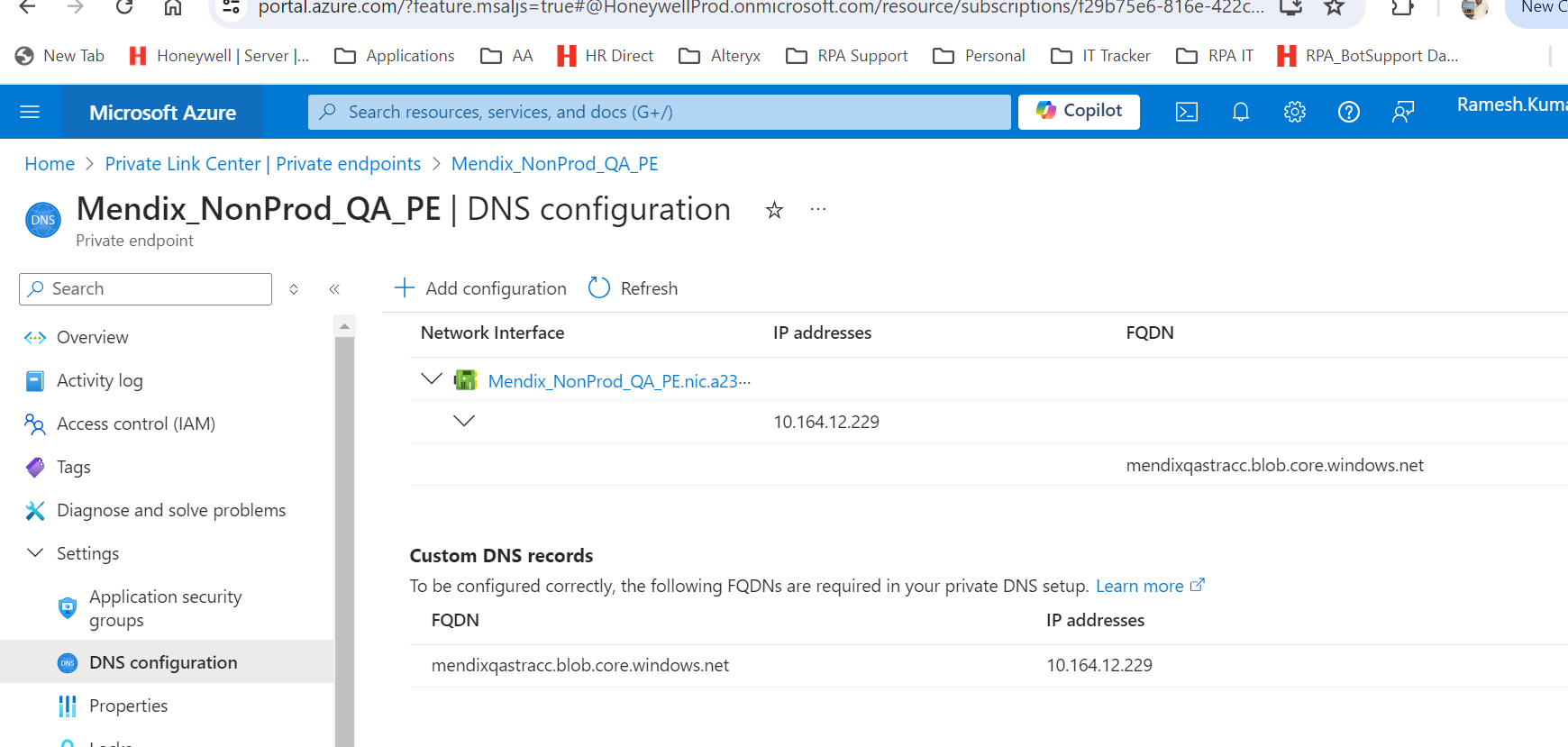
Enable support for customer-managed keys Blobs and files only

Enable infrastructure encryption Disabled

Create RITM Request to add private dns zone - [Link](https://honeywell.service-now.com/itdirect?id=sc_cat_item&sys_id=4b9e3a8b1b886190aaf2411abc4bcb2a)

Provide One of pe-subnet IP address & Record set name.

Validate after creation



# 11. Create SQL Managed Instance

Create a request to the DB team to create a SQL Managed Instance which will be used to configure the Database Plan for the Mendix setup.

Please fill in the DB intake form for Azure cloud. Do raise a normal change ticket and assign the ticket to “DL-SQL DB Non-Export Control [DL-SQLDBNon-ExportControl@HoneywellProd.onmicrosoft.com](mailto:DL-SQLDBNon-ExportControl@HoneywellProd.onmicrosoft.com)”

**Note:** Databases for Dev, QA and Prod are already created so ignore the creation.

**DEV:**

Instance Name: **corpsql4d.3d7f657415cf.database.windows.net**

DB Name: DB\_PLATFORM\_MENDIX\_DEV

Access provided to Account: [SV\_Mendix\_Pltfrm\_Dv@honeywell.com](mailto:SV_Mendix_Pltfrm_Dv@honeywell.com)

**QA:**

Instance Name: **corpsql4d.3d7f657415cf.database.windows.net**

DB Name: DB\_PLATFORM\_MENDIX\_QA

Access provided to Account**:** [SV\_Mendix\_Pltfrm\_QA@honeywell.com](mailto:SV_Mendix_Pltfrm_QA@honeywell.com)

username: sql\_mendix\_qa

password: MendixQa@123

**PROD:**

Instance Name: corpsql10p.7255dc89ed49.database.windows.net

DB Name: DB\_PLATFORM\_MENDIX\_PROD

Account: SV\_Mendix\_Pltfrm\_Prd@honeywell.com

**Important Note:** Azure AD authentication is **not a supported authentication method** at this time. In the next Mendix operator release we will be supporting workload identities in Azure and that will remove the dependency on using username and password.

# 12. Mendix for Private Cloud Setup

Mendix for Private Cloud allows you to deploy and manage your Mendix apps in a Kubernetes private cloud cluster.

You can use the **standalone** Mendix Operator to deploy Mendix apps through your own DevOps process, which is particularly useful for private clouds with an air-gap isolating them from the internet.

Refer this [link](https://docs.mendix.com/developerportal/deploy/private-cloud/#22-standalone-architecture) for standalone architecture.

Follow [link](https://docs.mendix.com/developerportal/deploy/private-cloud-cluster/) to Creating a Private Cloud Cluster

To allow you to manage the deployment of your apps to Kubernetes, you first need to create a cluster and add at least one namespace in the Mendix Portal.

## Creating a Cluster and Namespace

Follow the section in the <https://docs.mendix.com/developerportal/deploy/private-cloud-cluster/#3-creating-a-cluster-and-namespace> and complete steps up to 3.2.2 Adding a Namespace for Standalone Cluster.

## Installing and Configuring the Mendix Operator

You can install and run the Mendix Operator in Standard mode. In Standard mode, it is installed separately for each namespace where a Mendix app is deployed.

Running the Mendix Operator in Standard Mode – Follow the link <https://docs.mendix.com/developerportal/deploy/standard-operator/>

Complete the Step 2 Downloading the Configuration Tool

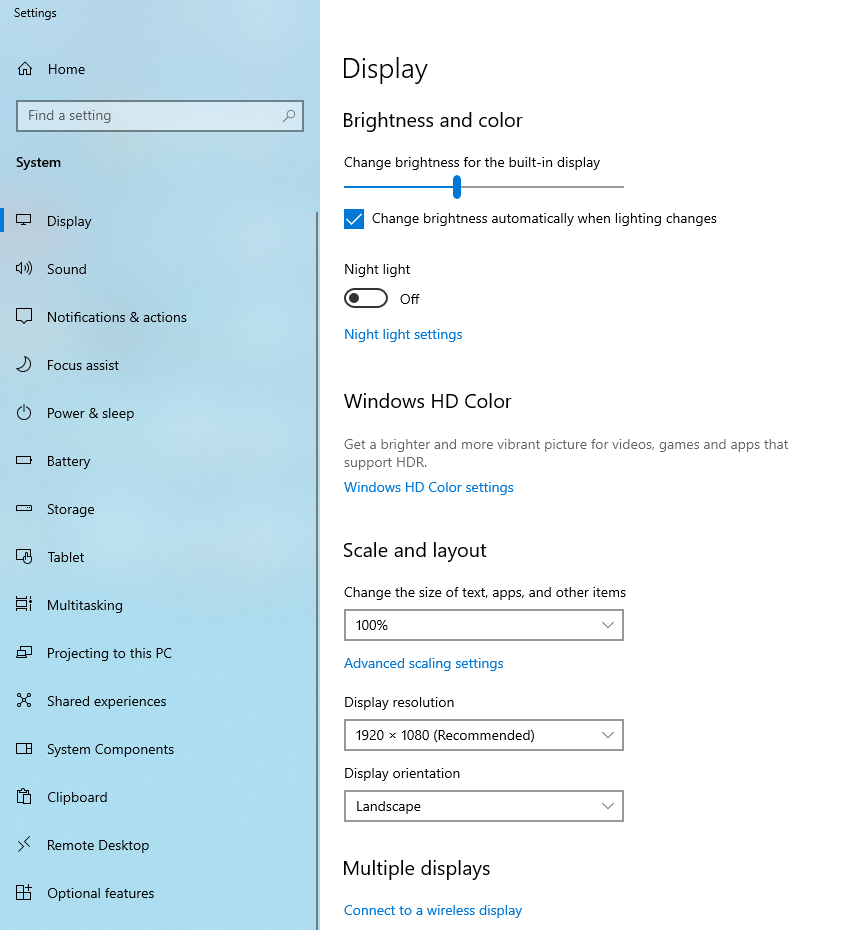
## Registry Configuration

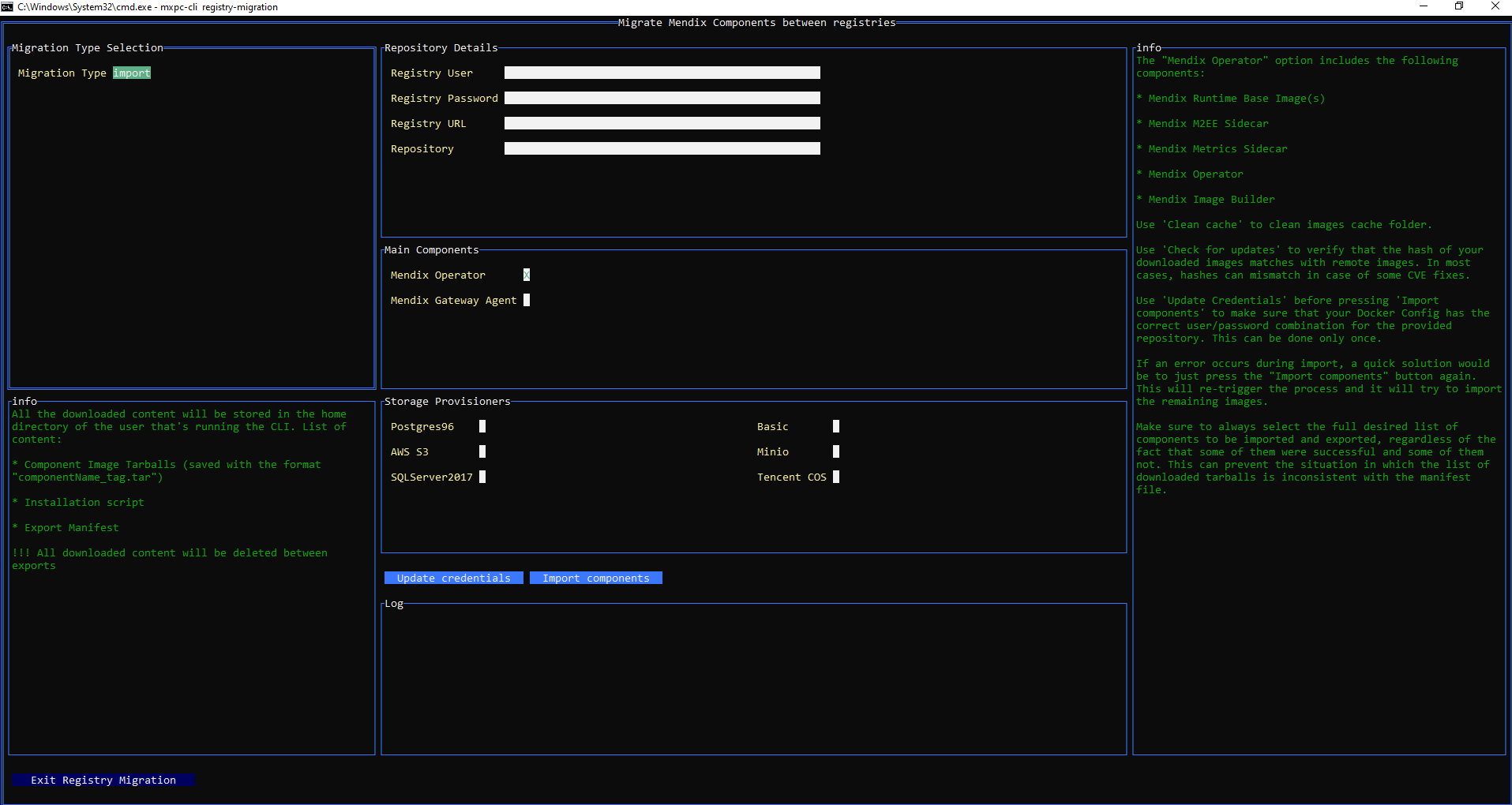
Refer link <https://docs.mendix.com/developerportal/deploy/private-cloud-registry/#24-azure-container-registry>

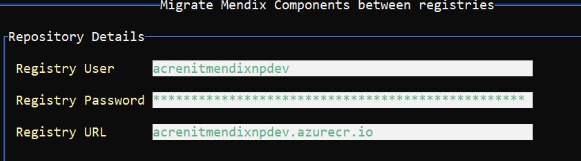
Run command from the local folder where the mxpc configuration tool is downloaded

mxpc-cli registry-migration

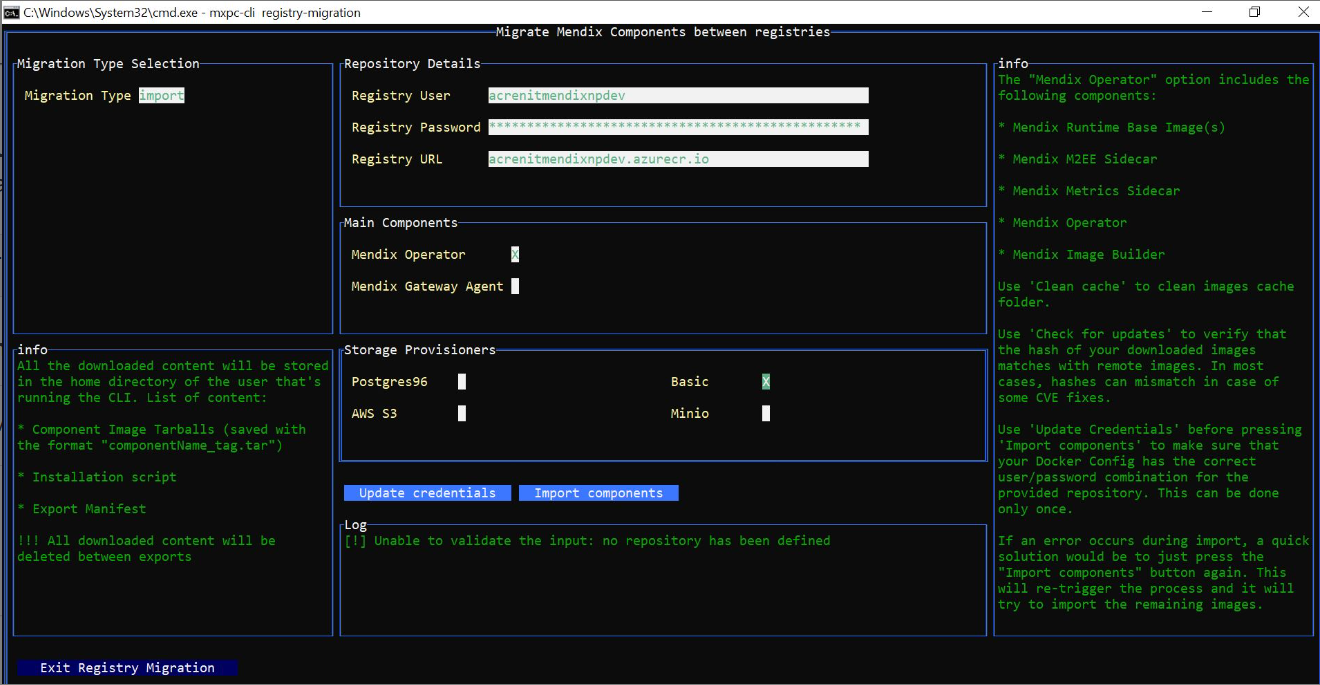
**Important Note:** There is a known issue with the mxpc configuration tool. Keep the computer/laptop display setting to the minimum to view full details in the tool. Make sure the Repository details have 4 input fields







Repository - acrenitmendixnpdev



## Running the mxpc Configuration Tool

Once you are signed in to your cluster you can run the Configuration Tool.

Run command from the local folder where the mxpc configuration tool is downloaded

.\mxpc-cli.exe installer -n mendixplatformqa

Refer the link https://docs.mendix.com/developerportal/deploy/standard-operator/#running-the-tool

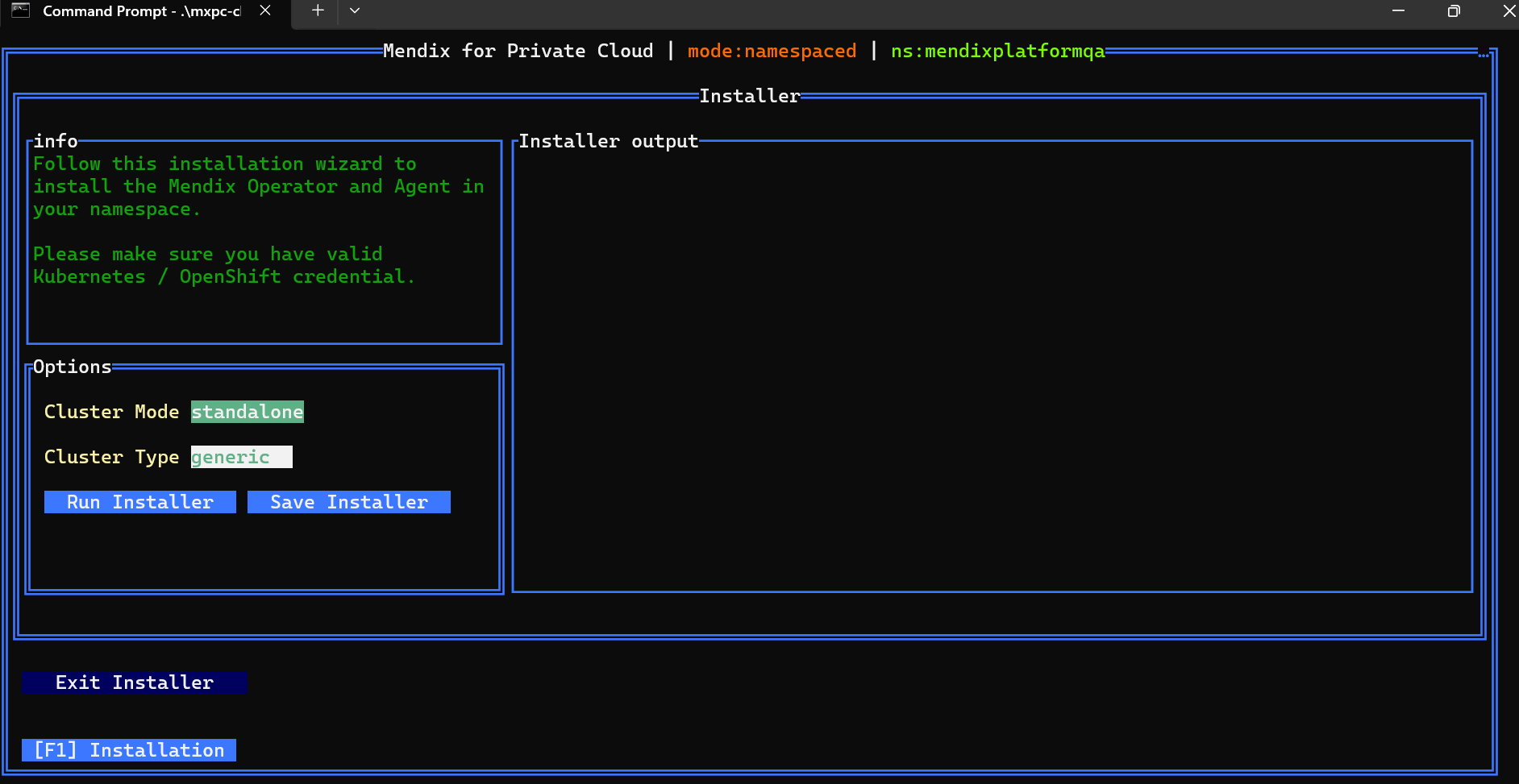
Complete all the steps mentioned except Step 4.4 Custom TLS

Use below images for different configuration setup that needs to be done.

Home page of installation



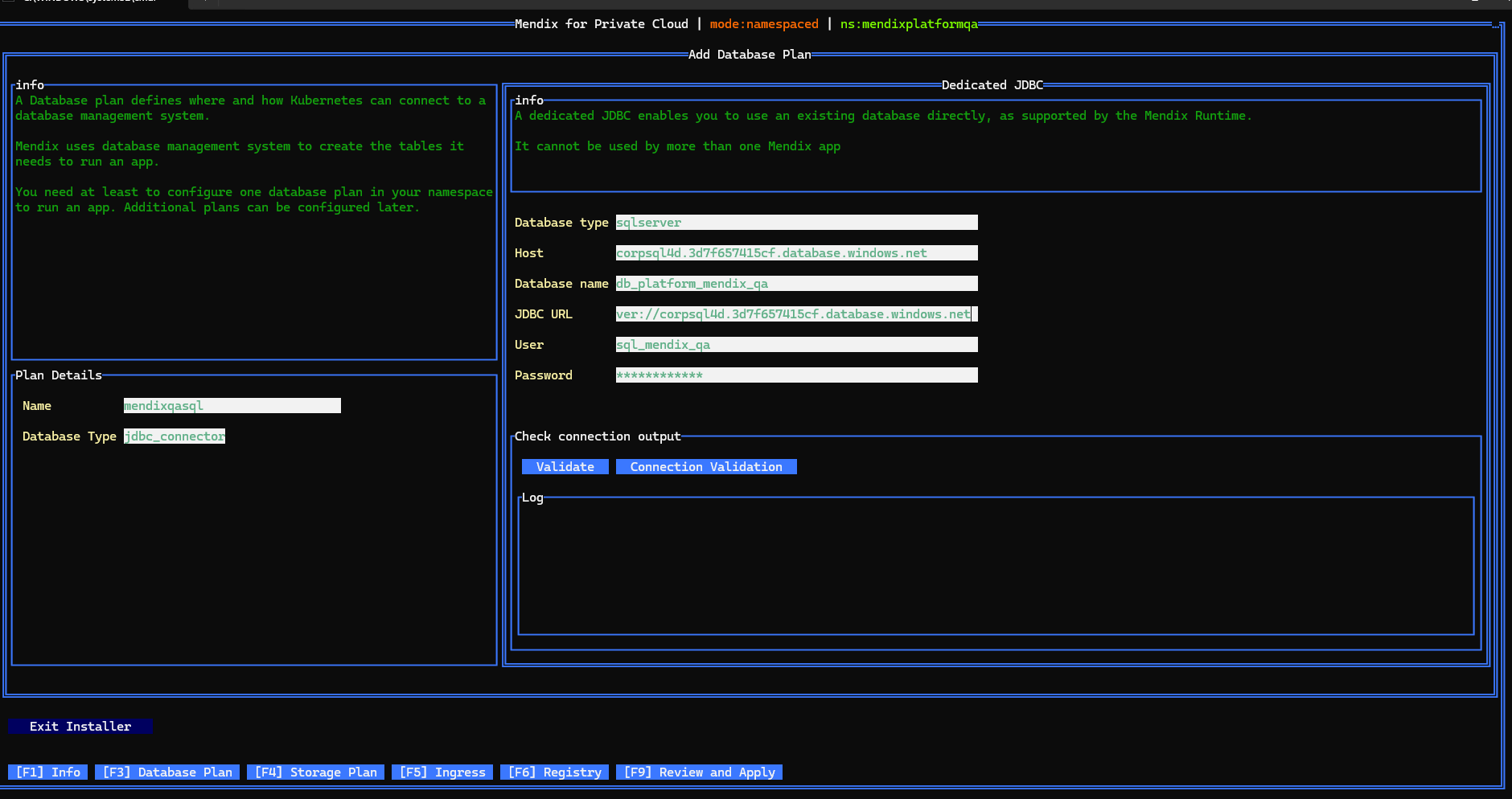
**Base Installation**:



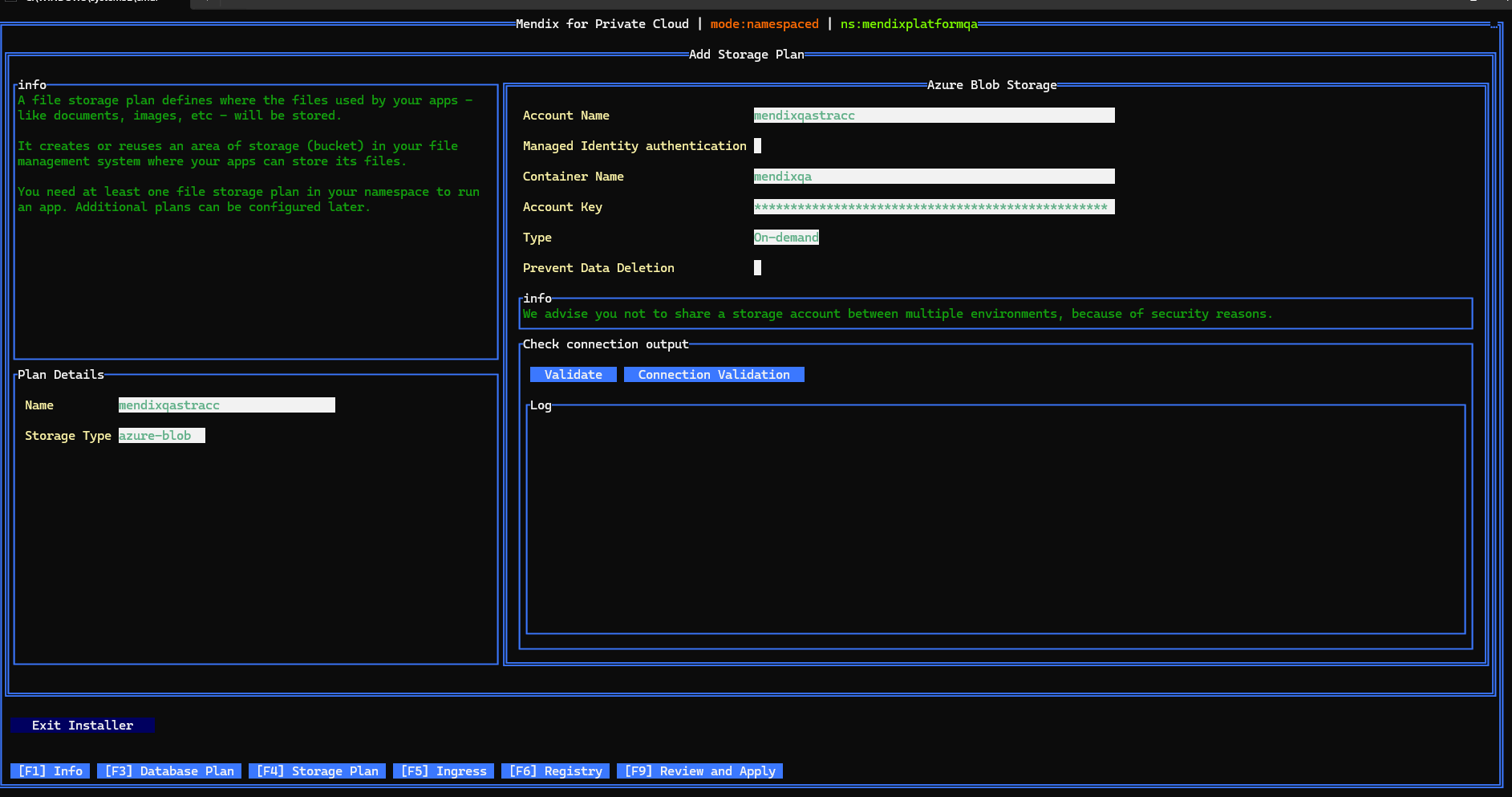
**Configure Namespace**



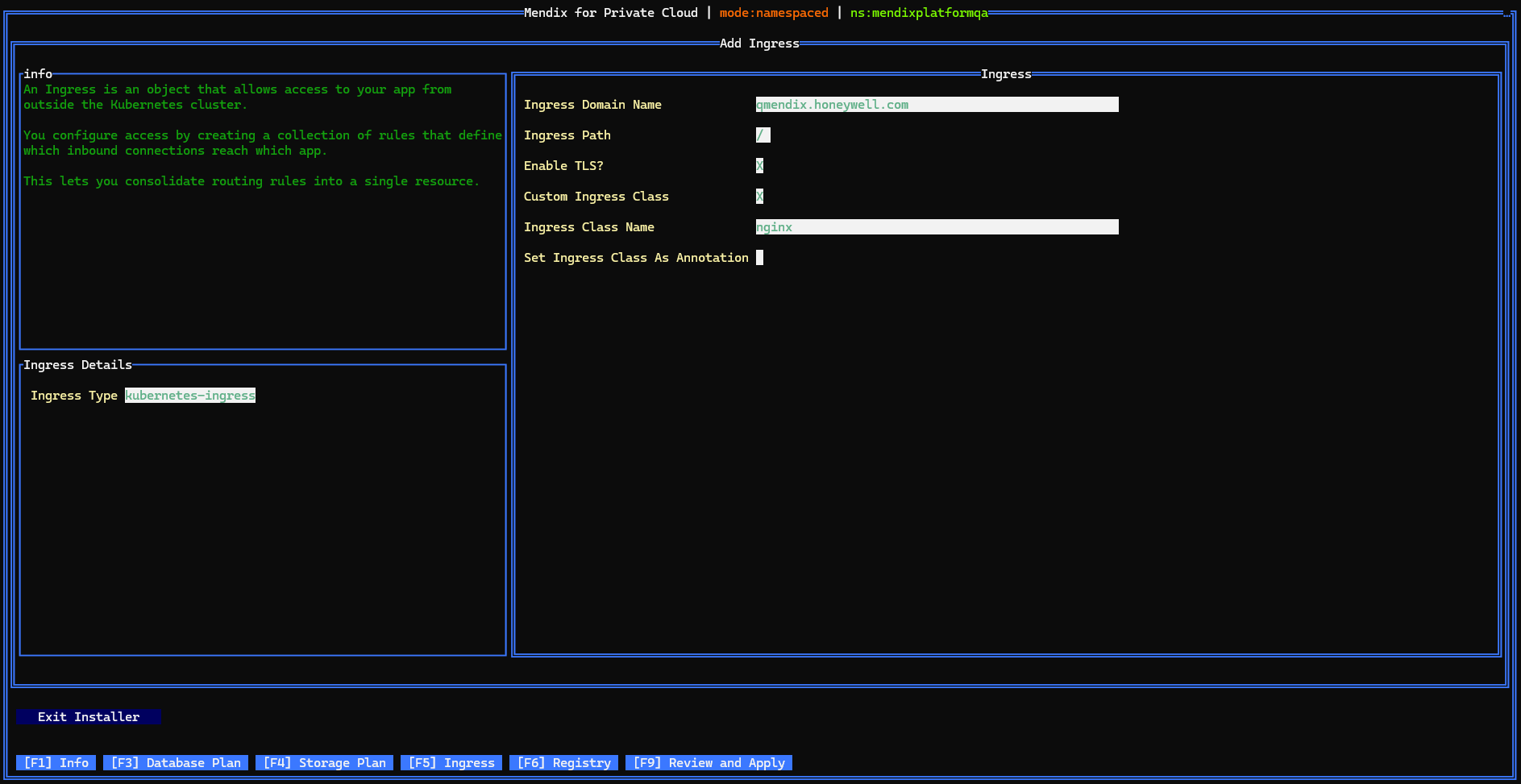
**Database Plan**



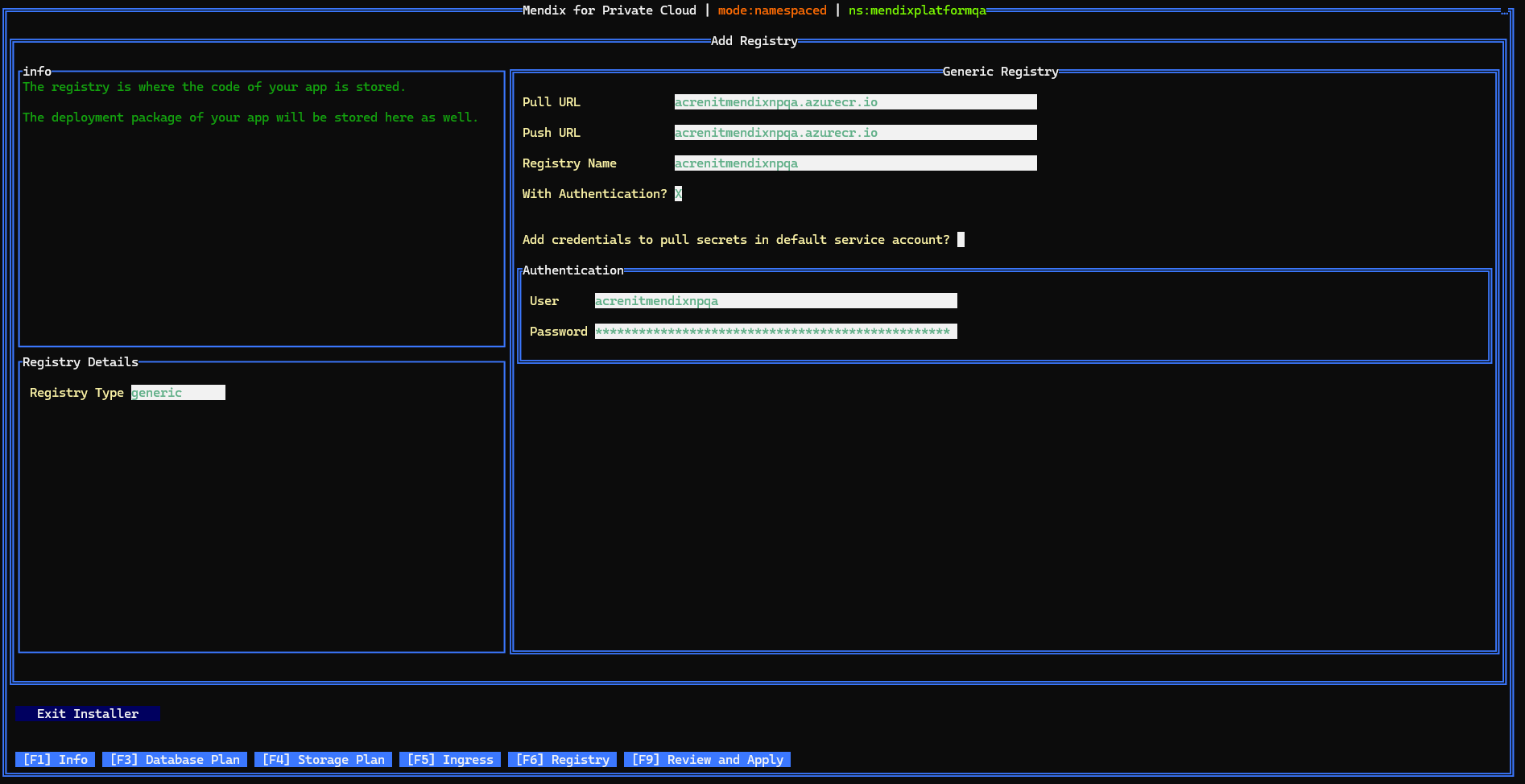
**Storage Plan**



**Ingress**



**Registry**



Once all the configurations are done click “Review and apply”

Generated Mendix configuration .yaml file is shown below

namespace: mendixplatformqa

cluster\_mode: standalone

mask:

database\_plan: true

storage\_plan: true

ingress: true

registry: true

proxy: false

custom\_tls: false

global\_operator: false

database\_plan:

name: mendixqasql

type: jdbc\_connector

jdbcdb:

host: corpsql4d.3d7f657415cf.database.windows.net

user: sql\_mendix\_qa

password: MendixQa@123

db\_name: db\_platform\_mendix\_qa

db\_type: sqlserver

jdbc\_url: jdbc:sqlserver://corpsql4d.3d7f657415cf.database.windows.net

storage\_plan:

name: mendixqastracc

type: azure-blob

azure:

reservation\_type: on-demand

container\_name: mendixqa

account\_name: mendixqastracc

account\_key: F78r8D9ZGEfsShdInikm+3sDC8yPZbqsk5a723cD1jmVpHLbqZgFAys2zC5FDbJPdPM8lU2vPjmS+AStRD1s0Q==

account\_subscription\_id: ""

account\_resource\_group: ""

azwi\_authentication: false

managed\_identity\_clientid: ""

kubernetes\_service\_account: ""

ingress:

type: kubernetes-ingress

enable\_tls: true

k8s\_ingress:

domain: qmendix.honeywell.com

useCustomIngressClass: false

customIngressClass: ""

customIngressClassAsAnnotation: false

path: /

service: null

registry:

type: generic

generic\_registry:

auth\_pull\_url: acrenitmendixnpqa.azurecr.io

auth\_push\_url: acrenitmendixnpqa.azurecr.io

registry\_name: acrenitmendixnpqa

enable\_auth: true

auth\_user: acrenitmendixnpqa

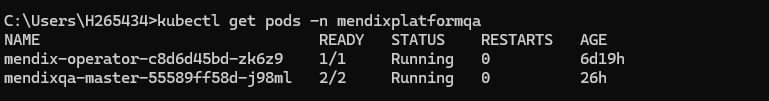
auth\_password: QySnmvGGW9XIeSEom7qyuTAIhXgmZ1e1HAS4ZNCSfj+ACRCeso4r

link\_secret\_to\_service\_account: false

## Verify Mendix Operator Running Status

Connect to the AKS cluster and run the command below and make sure the Operator and Master are in “Running” status.

kubectl get pods -n mendixplatformqa



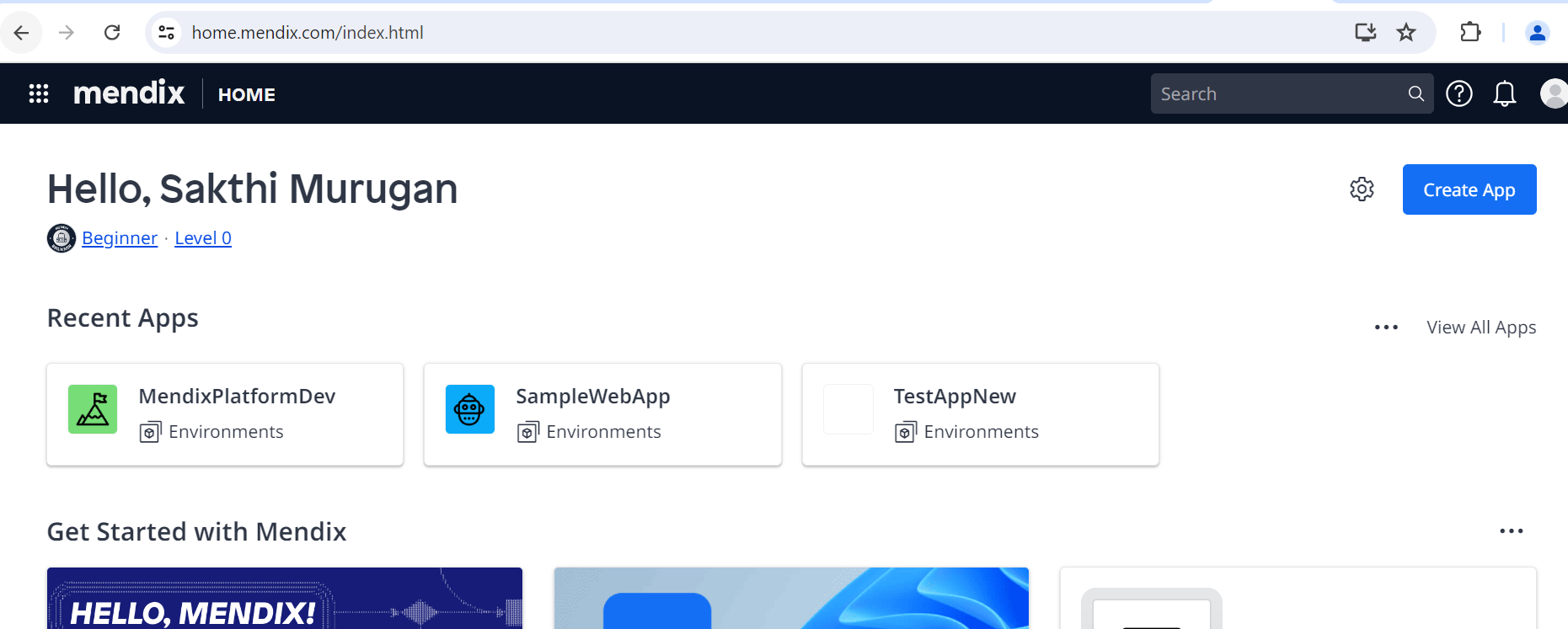
## Mendix Portal

Mendix portal(<https://home.mendix.com/>) is where we create and manage the Mendix applications.

### Register in Mendix Portal

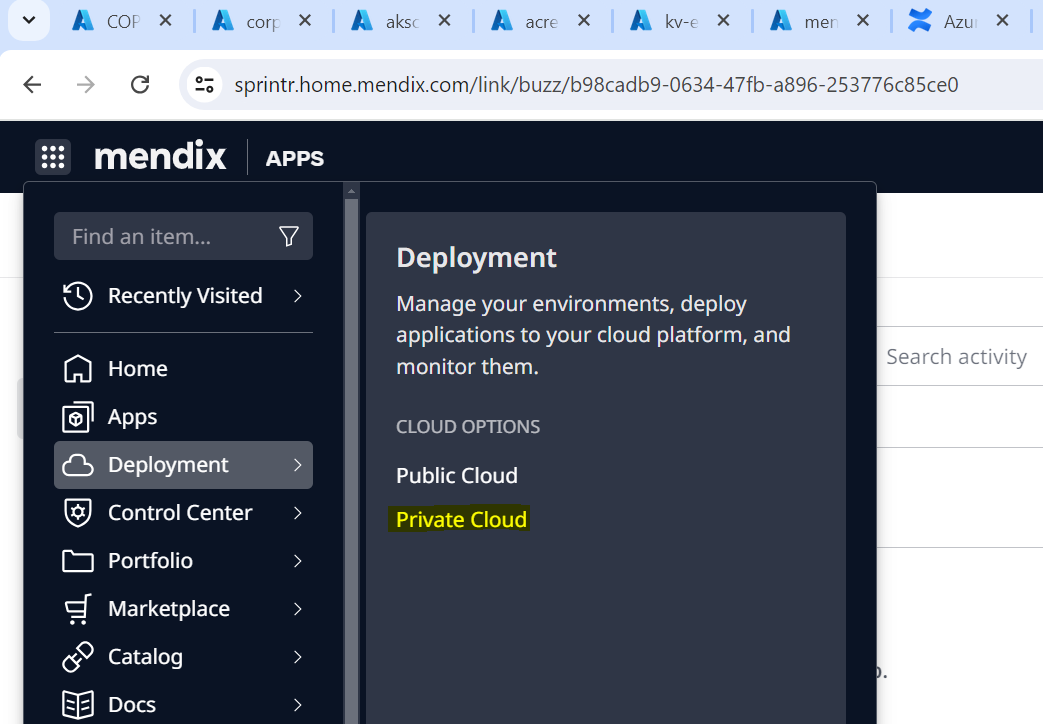
Register in the Mendix portal with your work email id and password.

After successful login you will see the home page as shown below



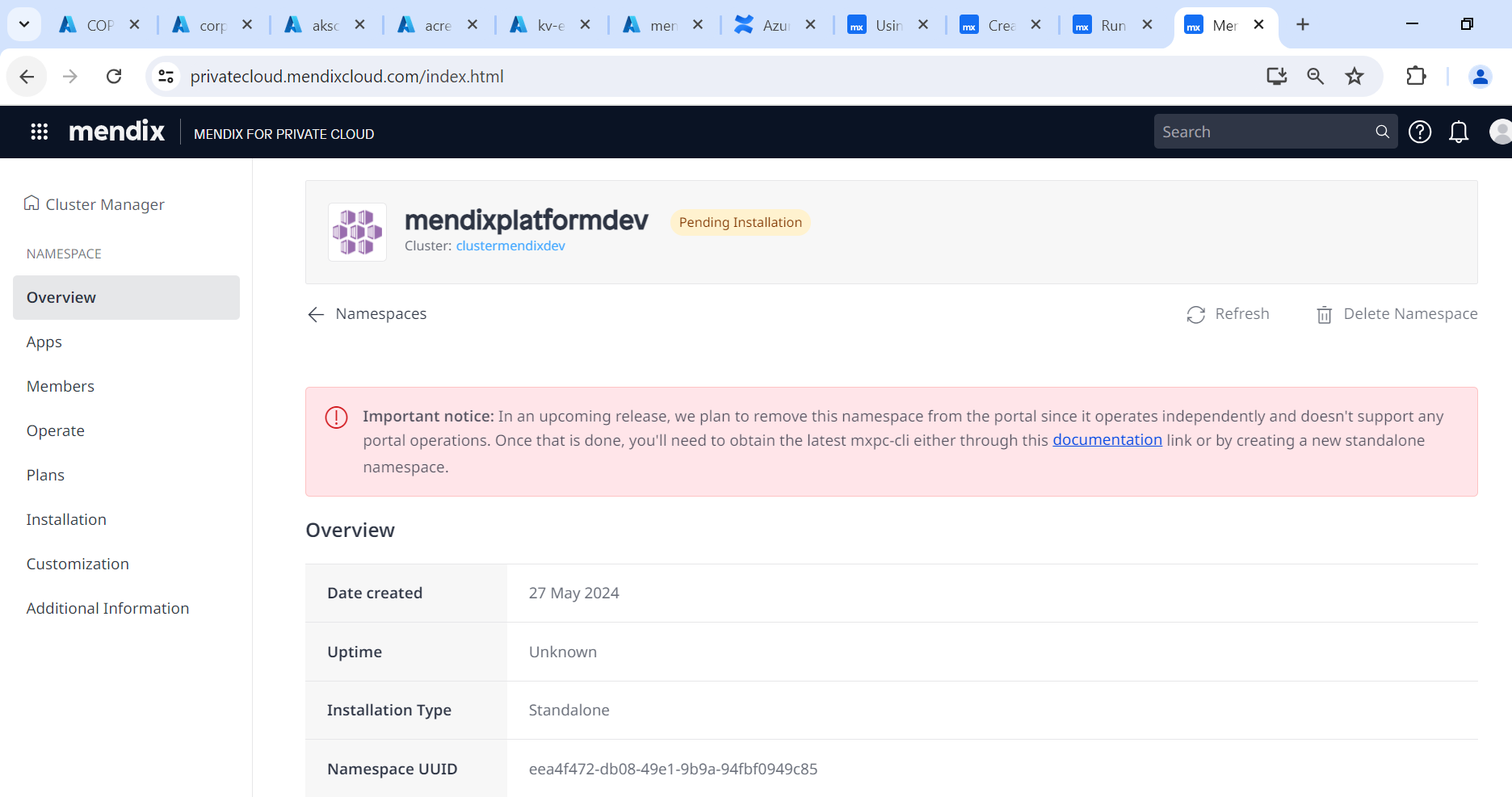
### 12.6.2 Access Private Cloud

Access your private cloud from here. <https://privatecloud.mendixcloud.com/index.html>



### 12.6.3 Private Cluster Details

Cluster details are shown as below



# 13. Deploy a Mendix App to a Private Cloud Cluster

Once you have the Mendix Operator installed in a namespace of your Kubernetes cluster, you can use it to control the deployment of your Mendix app using Mendix Custom Resources (CRs). The Mendix operator then creates the app container and builds the app inside the namespace, together with all the resources the app needs.

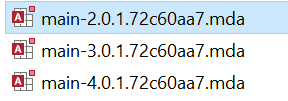
This [link](https://docs.mendix.com/developerportal/deploy/private-cloud-operator/) explains how to provide the CRs through the console or command line for a standalone cluster.

## 13.1 Deploying a Mendix App with an Operator

You can deploy multiple Mendix apps to run in the same Kubernetes namespace. Apps will have an Environment UUID added when they are deployed to ensure that they are unique in the project; the name is required to identify the app when creating, modifying, or deleting it.

Follow the instructions below to deploy your app.

Complete the step 3.1 Creating a Deployment Package. You will generate a .mda file



Complete the step 3.2 Editing the CR file 

The CR yaml file used for deployment of sample Mendix app is shown below

apiVersion: privatecloud.mendix.com/v1alpha1

kind: MendixApp

metadata:

name: mendixqa

spec:

database: # Specification of Database CR

servicePlan: upsell-jdbc-plan

storage: # Specification of Storage CR

servicePlan: mendixqastracc

# mendixRuntimeVersion: 10.12.1 # Studio Pro version of the Mendix app

sourceURL: https://mendixqastracc.blob.core.windows.net/mendixqa/main-4.0.1.72c60aa7.mda?sp=r&st=2024-07-30T09:20:17Z&se=2024-07-30T17:20:17Z&spr=https&sv=2022-11-02&sr=b&sig=XmR5GnxTTeeWiI3pYB5iNqAPXXxwTYvGteTOFDddaVs%3D # URL of App's source MDA or prebuilt OCI image

appURL: qmendix.honeywell.com # URL to access the app - example-mendixapp.k8s-cluster.example.com

tls: # Optional, can be omitted : set a custom TLS configuration, overriding the default operator configuration

# Enable or disable TLS for the app

enableTLS: true

# Optional: name of an existing kubernetes.io/tls secret containing the TLS certificate

secretName: upsell-mendixqa-tls-secret

replicas: 1 # Number of replicas, set to 0 to stop all replicas

runtime: # Configuration of the Mendix Runtime

runtimeLicense: # Mendix Runtime License configuration

type: offline # Set to offline

id: 2d2062d8-9921-457c-9fb3-647007b5e25e # Offline LicenseId (UUID) value provided by Mendix Support

key: 

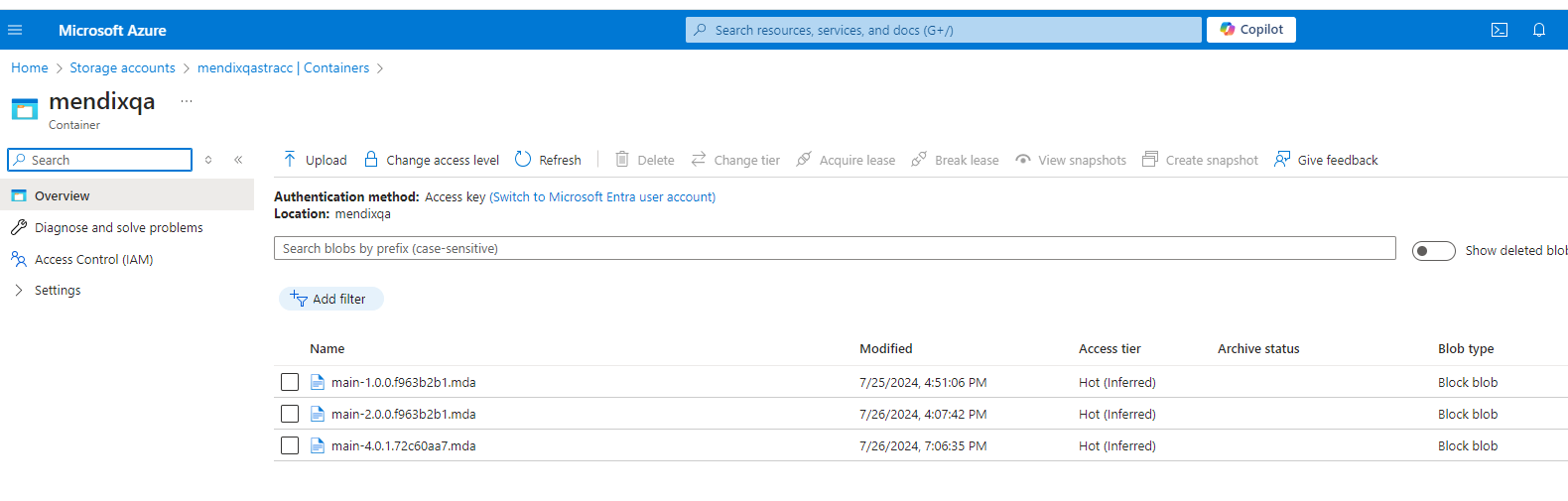
logAutosubscribeLevel: INFO # Default logging level

mxAdminPassword: TTNuZGl4QzAzQGRtaW5RQA== # base64 encoded password for MxAdmin user. In this example, 'M3ndixC03@dminQ@'; can be left empty keep password unchanged

debuggerPassword: V2VsYzBtZSE= # base64 encoded password for debuggerPassword. In this example, 'Welc0me!';

dtapMode: P # Security & runtime mode: P for production, D for development

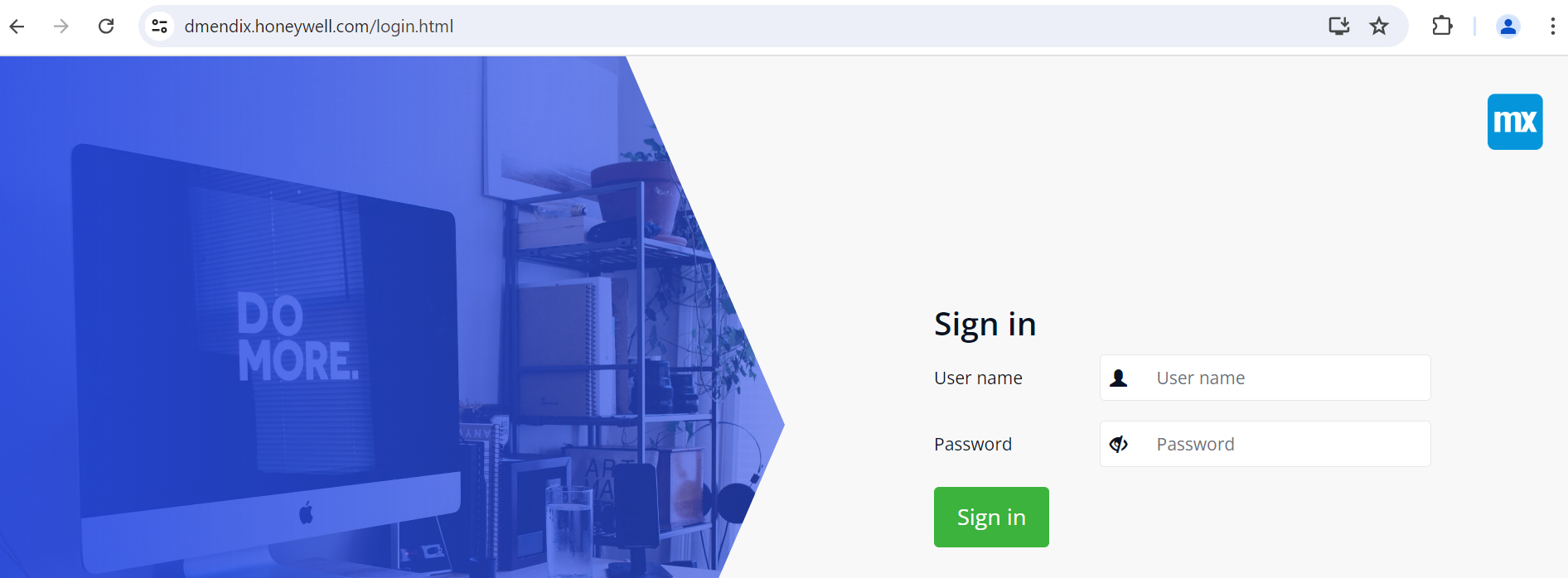
**Note:** The sourceURL must be a valid http/https url. So the .mda file is manually uploaded to the storage account(mendixdevstracc) and accessed using a SAS url.



Ignore steps.

* 3.2.1 Setting App Constants
* 3.2.2 Configuring Scheduled Events

## 13.2 Verify the Mendix App Is Running

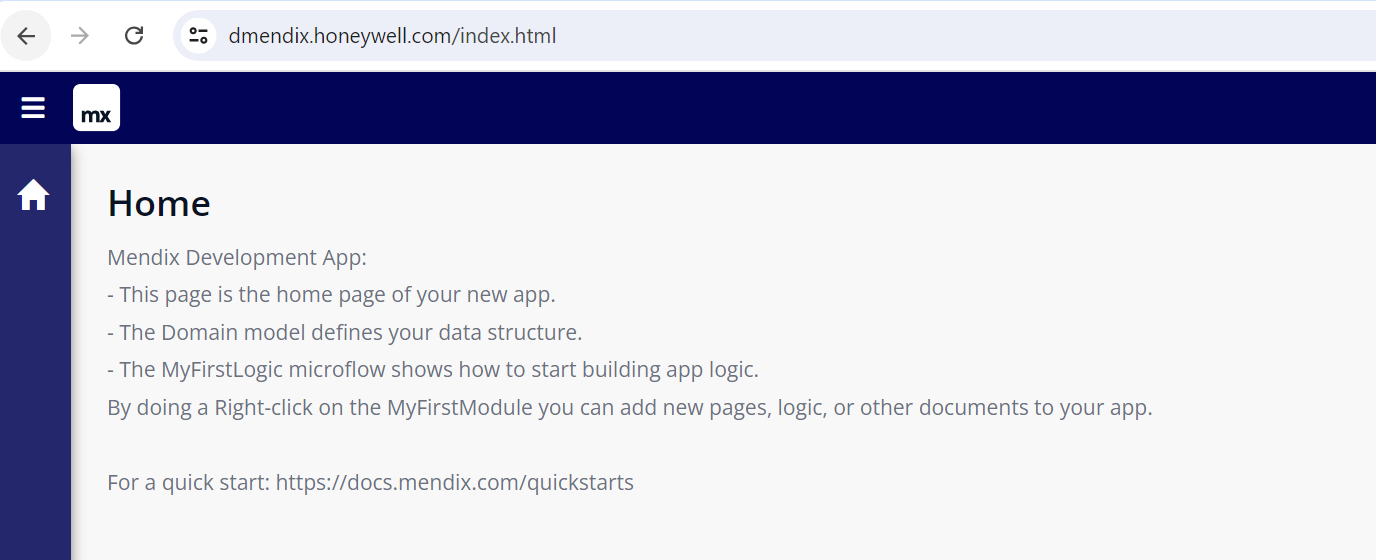
The app url is mentioned as <https://qmendix.honeywell.com/>. Browse the url to see the login page 

Login Credentials (as configured in CR file)

User Name – MxAdmin

Password - 'M3ndixC03@dminQ@

After successful login the App home page is displayed.



# 14. Mendix Academy

Here is the [Mendix Academy Learning Path to set up **Mendix for Private Cloud**](https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Facademy.mendix.com%2Flink%2Fpaths%2F101%2FMendix-for-Private-Cloud&data=05%7C02%7CSakthi.Murugan%40Honeywell.com%7C07e7ecaebd1248ba536f08dc915f6720%7C96ece5269c7d48b08daf8b93c90a5d18%7C0%7C0%7C638545086159662005%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C0%7C%7C%7C&sdata=64A5%2BPB7kyLNTQAmrtL7sWVJF4zJwPZtT2NIryPQbdE%3D&reserved=0)

# 15. Mendix Support Ticket

You can raise support ticket to Mendix using the link [https://support.mendix.com](https://nam12.safelinks.protection.outlook.com/?url=https%3A%2F%2Fsupport.mendix.com%2Fhc%2Frequests%2F218633&data=05%7C02%7Csakthi.murugan%40honeywell.com%7Ce89461461ee94d5477b208dc83bb89a2%7C96ece5269c7d48b08daf8b93c90a5d18%7C0%7C0%7C638530088713131415%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C0%7C%7C%7C&sdata=l8iAEJHPX2OMB0jlTKZpaEu9%2BNgdyAnoUgMxw3w5%2B%2BQ%3D&reserved=0)

# 16. Mendix Installation Troubleshooting Steps

## 16.1 General Check On Pods and Logs

kubectl get pods

kubectl logs <operator podname>

kubectl get mendixapp

kubectl describe mendixapp <mendixappname>

Example: kubectl describe mendixapp mendixqa

[mendix qa metadata field in CR file]

## 16.2 Networking Issue

kubectl get ingress

kubectl describe ingress if networking issue

## 16.3 App Issue

Kubectl logs <mendix master podname> - Gives App Errors

## 16.4 Deployment Issue

No Master Pod

No Error in operator podname

kubectl logs mendixdev-master-fff456448-kfzvg

kubectl logs mendixdev-master-fff456448-kfzvg m2ee-sidecar

1/2 running issue

ideal 2/2 running must run

Scenarios - production not enabled

exception failed to start

## 16.5 Port Forwarding To Test Mendix App in localhost

kubectl port-forward service/mendixqa 8080:8080 -n mendixplatformqa

Then check the app is running in the url <https://localhost:8080>