Problem Statement :

Deploying a multi-container application to Azure Kubernetes Services

Azure Kubernetes Service (AKS)is the quickest way to use Kubernetes on Azure.**Azure Kubernetes Service (AKS)**manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. Azure DevOps helps in creating Docker images for faster deployments and reliability using the continuous build option.

One of the biggest advantage to use AKS is that instead of creating resources in cloud you can create resources and infrastructure inside Azure Kubernetes Cluster through Deployments and Services manifest files.

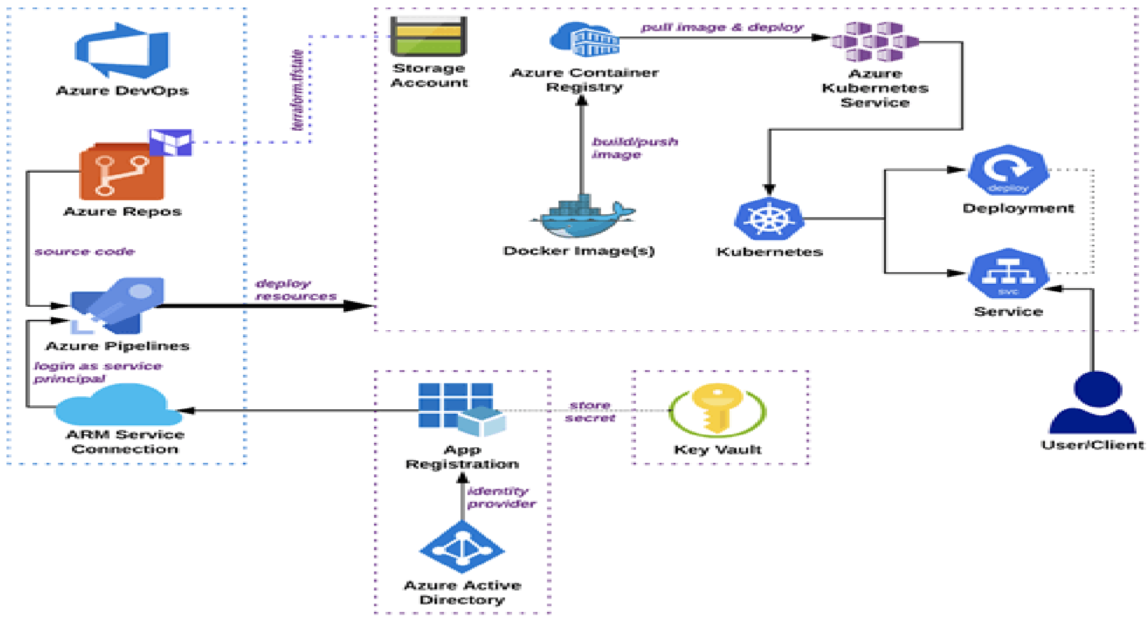
Lab Scenario

This lab uses a Dockerized ASP.NET Core web application -**MyHealthClinic**(MHC) and is deployed to a**Kubernetes**cluster running on**Azure Kubernetes Service (AKS)** using **Azure DevOps**.

There is a**mhc-aks.yaml** manifest file which consists of definitions to spin up Deployments and Services such as **Load Balancer** in the front and **Redis Cache** in the backend. The MHC application will be running in the mhc-front pod along with the Load Balancer.

The following image will walk you through all the steps explained in this lab:

Solution/ Architecture



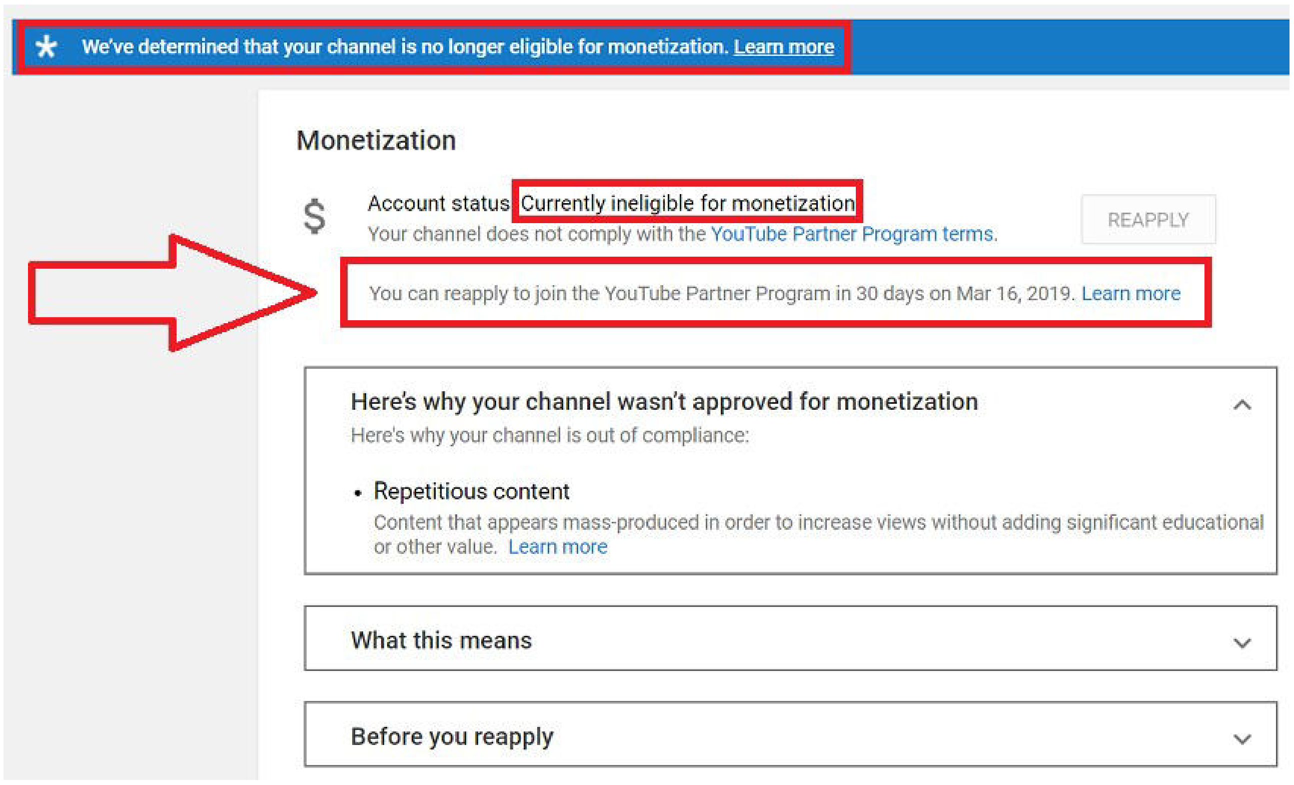
The following tasks will be performed:

•Create an Azure Container Registry (ACR), AKS and Azure SQL server

•Provision the Azure DevOps Team Project with a .NET Core application using the Azure DevOps Demo Generator tool.

•Configure application and database deployment, using Continuous Deployment (CD) in the Azure DevOps

•Initiate the build to automatically deploy the application



We've determined that your channel is no longer eligible for monetization. Learn more

Monetization

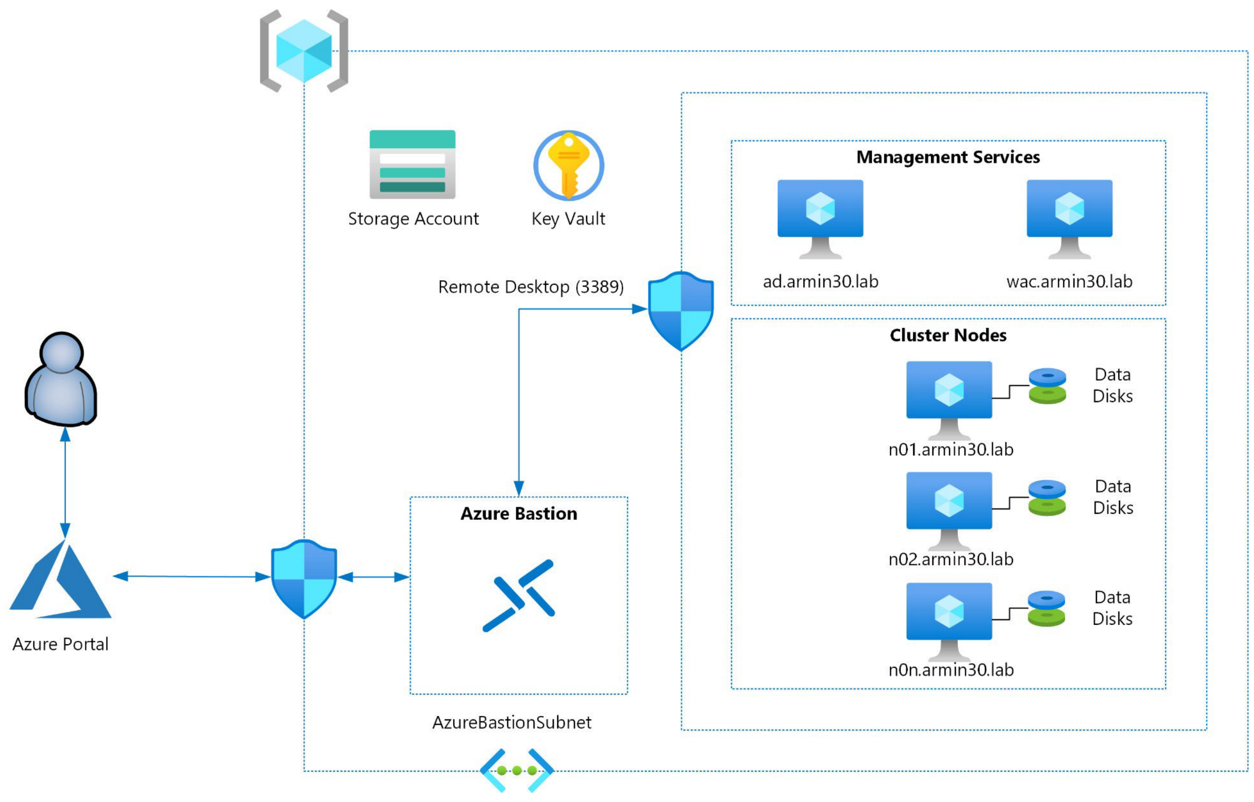
Account status Currently ineligible for monetization
Your channel does not comply with the YouTube Partner Program terms.

You can reapply to join the YouTube Partner Program in 30 days on Mar 16, 2019. Learn more

Here's why your channel wasn't approved for monetization
Here's why your channel is out of compliance:

- Repetitious content
  Content that appears mass-produced in order to increase views without adding significant educational or other value. Learn more

Technical Details and Implementation of solution



Launch theAzure Cloud Shellfrom the Azure portal and choose**Bash**.

1.**Deploy Kubernetes to Azure, using CLI**:

i. Get the latest available Kubernetes version in your preferred region into a bash variable. Replace`<region>`with the region of your choosing, for example eastus.

```
  version=$(az aks get-versions -l <region> --query
'orchestrators[-1].orchestratorVersion' -o tsv)
```

ii. Create a Resource Group

```
  az group create --name akshandsonlab --location <region>
```

iii. Create AKS using the latest version available

```
 az aks create --resource-group akshandsonlab --name <unique-aks-
cluster-name> --enable-addons monitoring --kubernetes-version
$version --generate-ssh-keys --location <region>
```

Deploy Azure Container Registry(ACR): Run the below command to create your own private container registry using Azure Container Registry (ACR).

```
 az acr create --resource-group akshandsonlab --name <unique-acr-name>
--sku Standard --location <region>
```

Authenticate with Azure Container Registry from Azure Kubernetes Service: When you're using Azure Container Registry (ACR) with Azure Kubernetes Service (AKS), an authentication mechanism needs to be_established. You can set up the AKS to ACR integration in a few simple commands with the Azure CLI. This integration assigns the

AcrPull role to the managed identity associated to the AKS Cluster. Replace the variables `$AKS_RESOURCE_GROUP, $AKS_CLUSTER_NAME, $ACR_NAME` with appropriate values below and run the command.

```
 az aks update -n $AKS_CLUSTER_NAME -g $AKS_RESOURCE_GROUP --
attach-acr $ACR_NAME
```

For more information see document on how to Authenticate with Azure Container Registry from Azure Kubernetes Service

1.**Create Azure SQL server and Database**: Create an Azure SQL server.

```
 az sql server create -l <region> -g akshandsonlab -n <unique-
sqlserver-name> -u sqladmin -p P2ssw0rd1234
```

Create a database

```
 az sql db create -g akshandsonlab -s <unique-sqlserver-name> -n
mhcdb --service-objective S0
```

Challenges in implementing the solution

You will be prompted to authorize this connection with Azure credentials. Disable pop-up blocker in your browser if you see a blank screen after clicking the OK button, and please retry the step.

This creates an **Azure Resource Manager Service Endpoint**, which defines and secures a connection to a Microsoft Azure subscription, using Service Principal Authentication (SPA). This endpoint will be used to connect **Azure DevOps** and **Azure**.

The Azure portal includes a Kubernetes resource viewer (preview) for easy access to the Kubernetes resources in your Azure Kubernetes Service (AKS) cluster. Viewing Kubernetes resources from the Azure portal reduces context switching between the Azure portal and the kubectl command-line tool, streamlining the experience for viewing and editing your Kubernetes resources. The resource viewer currently includes multiple resource types, such as deployments, pods, and replica sets.

The Kubernetes resource view from the Azure portal replaces the AKS dashboard add-on, which is set for deprecation.

Business Benefit

It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline. Azure DevOps helps in creating Docker images for faster deployments and reliability using the continuous build option.