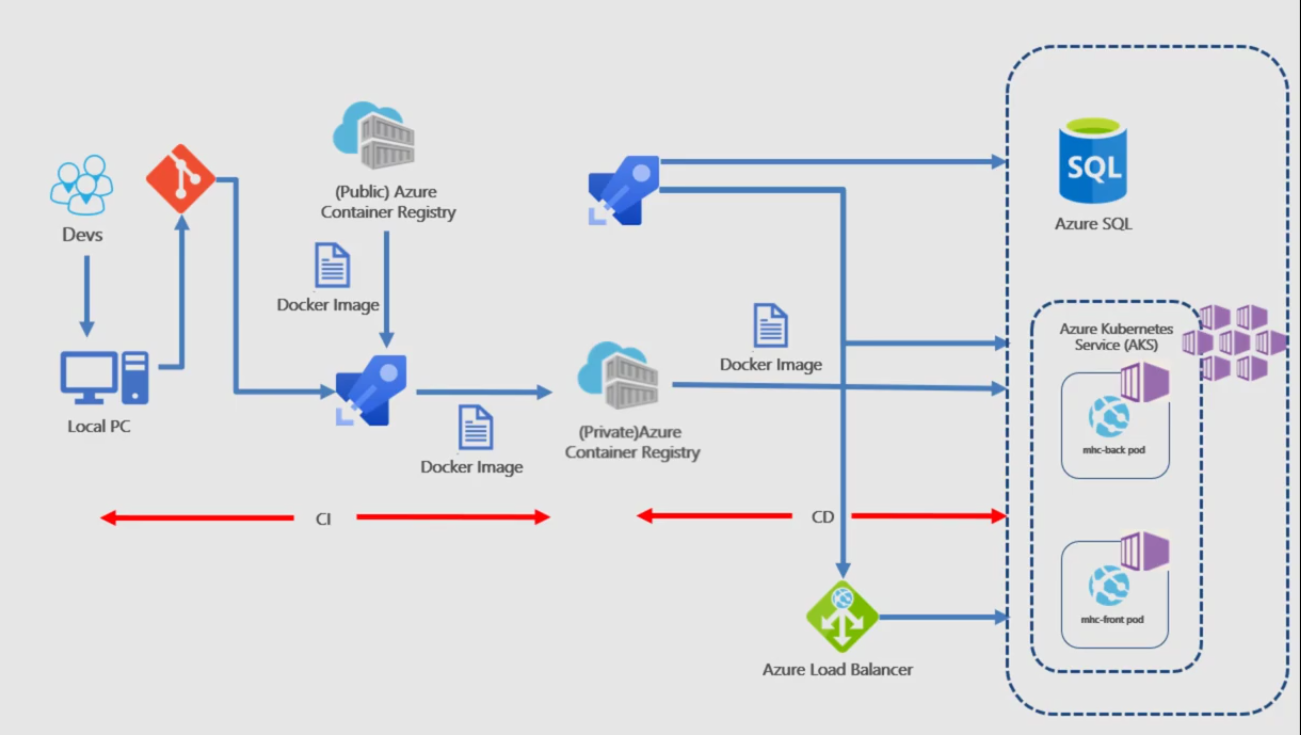
**Deploying a multi-container application to Azure Kubernetes Services**

**Scenario:** A Dockerized ASP.NET Core web application - MyHealthClinic (MHC) is used and deployed to a Kubernetes cluster running on Azure Kubernetes Service (AKS) using Azure DevOps.

A mhc-aks.yaml manifest file consists of 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.



**Setting up the environment**

Azure Container Registry Used to store the Docker images privately

AKS Docker images are deployed to Pods running inside AKS

Azure SQL Server SQL Server on Azure to host database

1. **Deploy Kubernetes to Azure, using CLI**:
2. Get the latest version of Kubernetes

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

1. Create a Resource Group

az group create --name akshandsonlab --location 'eastus'

1. Create AKS using the latest version available

az aks create --resource-group akshandsonlab --name agsaks1 --enable-addons monitoring --kubernetes-version $version --generate-ssh-keys --location 'eastus'

1. **Deploy Azure Container Registry(ACR)**:

create your own private container registry using Azure Container Registry (ACR)

az acr create --resource-group akshandsonlab --name agsacr1 --sku Standard --location eastus

**Grant AKS-generated Service Principal access to ACR** :

Authorize aks cluster to connect to ACR using aks service principal

1. Get the id of the service principal configured for AKS

CLIENT\_ID=$(az aks show --resource-group akshandsonlab --name agsaks1 --query "servicePrincipalProfile.clientId" --output tsv)

1. Get the ACR registry resource id

ACR\_ID=$(az acr show --name agsacr1 --resource-group akshandsonlab --query "id" --output tsv)

1. Create role assignment

az role assignment create --assignee $CLIENT\_ID --role acrpull --scope $ACR\_ID

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

az sql server create -l eastus -g akshandsonlab -n agssqlserver1 -u sqladmin -p P2ssw0rd1234

1. Create a database

az sql db create -g akshandsonlab -s agssqlserver1 -n agsdb1 --service-objective S0

**Configure Build and Release pipeline**

Here we manually map Azure resources such as AKS and Azure Container Registry to the build and release definitions in the AKS project in Azure DevOps organization.

**In the Build pipeline**

In **Run services, Build services, Push services** and **Lock services** tasks, select your Azure subscription from **Azure subscription** dropdown. Click **Authorize**.

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**.

Select appropriate values from the dropdown for - **Azure subscription** and **Azure Container Registry**

**Replace tokens** replace ACR in **mhc-aks.yaml** and database connection string in **appsettings.json**

**Run services** prepares suitable environment by pulling required image such as aspnetcore-build:1.0-2.0 and restoring packages mentioned in **.csproj**

**Build services** builds the docker images specified in a **docker-compose.yml** file and tags images with **$(Build.BuildId)** and **latest**

**Push services** pushes the docker image **myhealth.web** to Azure Container Registry

**Publish Build Artifacts** publishes **mhc-aks.yaml** & **myhealth.dacpac** files to artifact drop location in Azure DevOps so that they can be utilized in Release Definition

**applicationsettings.json** file contains details of the database connection string used to connect to Azure database which was created in the beginning of this lab.

**mhc-aks.yaml** manifest file contains configuration details of **deployments**, **services** and **pods**  which will be deployed in Azure Kubernetes Service.

Click on the **Variables** tab and Update **ACR** and **SQLserver** values for **Pipeline Variables** with the details noted earlier while configuring the environment.

**In the release Pipeline,**

In the **Dev** environment, under the **DB deployment** phase, select **Azure Resource Manager** from the drop down for **Azure Service Connection Type**, update the **Azure Subscription** value from the dropdown for **Execute Azure SQL: DacpacTask** task.

In the **AKS deployment** phase, select **Create Deployments & Services in AKS** task.

Update the **Azure Subscription**, **Resource Group** and **Kubernetes cluster** from the dropdown. Expand the **Secrets** section and update the parameters for **Azure subscription** and **Azure container registry** from the dropdown.

Repeat similar steps for **Update image in AKS** task.

Select the **Variables** section under the release definition, update **ACR** and **SQLserver** values for **Pipeline Variables** with the details noted earlier while configuring the environment. Select the **Save** button.

**Trigger a Build and deploy application**

**In the Build Pipeline**

Trigger the build manually and the build will generate and push the docker image to ACR. After the build is completed, you will see the build summary. To view the generated images navigate to the Azure Portal, select the **Azure Container Registry** and navigate to the **Repositories**.

**In the Release Pipeline**

Double-click on the latest release and select **In progress** link to see the live logs and release summary.

Once the release is complete, launch the [Azure Cloud Shell](https://docs.microsoft.com/en-in/azure/cloud-shell/overview) and run the below commands to see the pods running in AKS:

**az aks get-credentials --resource-group akshandsonlab --name agsaks1**

**kubectl get pods**

**kubectl get service mhc-front –watch**

Copy the **External-IP** and paste it in the browser and press the Enter button to launch the application