

Lab1 - Deploying a multi-container application to Azure Kubernetes Service

Overview

Azure Kubernetes Service (AKS) is the quickest way to use Kubernetes on Azure. 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 advantages to using AKS is that it is a managed environment. Instead of directly creating virtual machines and installing Kubernetes in the cloud (which you can do) you can specify key parameters for your Kubernetes cluster, and then deploy your containerized app to the Azure-managed 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 AKS using Azure DevOps. The lab will illustrate the use of both Kubectl imperative management and YAML file declarative management, but you won't have to directly build any YAML files (unless you want to).

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 lab also uses the **Azure DevOps Demo Generator** to autogenerate and populate your Azure DevOps project in your Azure DevOps organization. This template-based content includes source code, work items, iterations, service endpoints, build and release definitions, and more! The original purpose of this system is to simplify working with the Azure DevOps hands-on-labs, demos and other education material. But it can also be used to drive your own Azure DevOps automation utilities, provision your own custom templates, or as a reference for using the Azure DevOps REST APIs.

What's covered in this lab

The following tasks will be performed:

- Create an AKS cluster that will host your ASP.NET Core web application.
- Create an Azure Container Registry (ACR) which will host your container images in a private repository and create an Azure SQL server instance.

- 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 Azure DevOps.
- Initiate the build to automatically deploy the application.

Prerequisites

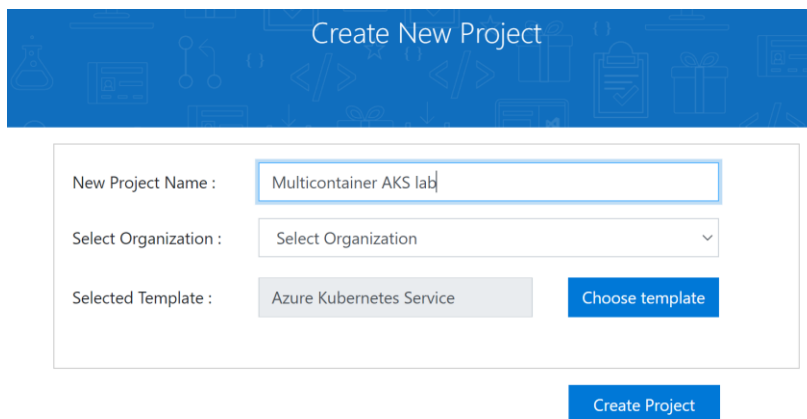
1. **Microsoft Azure Account:** You will need a valid and active Azure account for the Azure labs. If you do not have one, you can sign up for a [free trial](#).
2. You will need an **Azure DevOps** account. If you do not have one, you can sign up for free [here](#)
3. Ensure your account has the following roles:
 - The [Owner](#) built-in role for the subscription you will use.
 - A [Member](#) user in the Azure AD tenant you will use. (Guest users will not have the necessary permissions).

Setting up the Azure DevOps Project

Azure DevOps Demo Generator helps you create team projects on your Azure DevOps Organization with sample content that include source code, work items, iterations, service endpoints, build and release definitions based on the template you choose during the configuration.

Complete the following after you have created your DevOps Organization.

Follow the [simple walkthrough](#) to create projects on your Azure DevOps Services organization with pre-populated sample content. For this lab the **Azure Kubernetes Service** template is used which is already selected when you click on the link above. Make sure to enter a unique Project Name, and select your previously created DevOps organization.



Create New Project

New Project Name : Multicontainer AKS lab

Select Organization : Select Organization

Selected Template : Azure Kubernetes Service Choose template

Create Project

There are some additional extensions required for this lab and can be automatically installed during the process. These may appear after you select your organization.

Verifying if all required extension(s) are installed and enabled

All required extensions are installed/enabled in your Azure DevOps Organization.


- ✓ **Kubernetes extension**
- ✓ **Replace Tokens**

Create Project

Click on the Create Project button to provision your new project. Once provisioned you should receive confirmation on the DevOps Demo Generator page, and receive an email confirming completion of the repo.

Congratulations! Your project is successfully provisioned.

Navigate to project

Like the tool? Share your feedback 

- ✓ Project Multicontainer AKS workshop lab 1 created
- ✓ 2 team(s) created
- ✓ Board-Column, Swimlanes, Styles updated
- ✓ Work Items created
- ✓ Build definition created
- ✓ Release definition created

Your import is complete.

Congratulations! Your request to import **AKS repository** has completed successfully.




Go to repo



Let's proceed to setting up the Azure environment for AKS provisioning.

Setting up the environment

The following azure resources need to be configured for this lab:

Azure resources	Description
 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

Azure Services Naming

The following services will require you to enter a name. The recommended naming conventions can be found [here](#).

1.Azure Region – To see a list of available regions use the following AZ CLI command. If not familiar with the Azure Cloud Shell you can see the details [here](#).

az account list-locations

2.Resource Group

3.AKS Cluster Name

4.Azure Container Registry (ACR)

5.SQL Server and Database

Provisioning Services

1. Launch the [Azure Cloud Shell](#) from **inside the Azure portal** and choose **Bash**. You will be using the Azure CLI in the next steps, and the CLI is already installed in the Azure Cloud Shell.
2. Create a Kubernetes cluster in Azure, using CLI. This approach is an imperative approach using Kubectl, which is already installed in the Azure Cloud Shell.

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**. Use the echo command to view the value of the Version variable.

```
version=$(az aks get-versions -l <region> --query 'orchestrators[0].orchestratorVersion' -o tsv)
```

```
echo $version
```

ii. Create a Resource Group – make sure to replace the **###** characters in the name parameter below with your initials to ensure uniqueness.

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

iii. Create the AKS cluster using the latest version available. Make sure to change the name for the cluster, and replace the placeholder **<unique-aks-cluster-name>** with your name. A complete list of the AZ CLI commands for AKS can be found [here](#). Take a quick look at a comprehensive list of the az aks create command parameters, noting the optional and required parameters.

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

Important: Enter a unique AKS cluster name. AKS name must contain between 3 and 31 characters inclusive. The name can contain only letters, numbers, and hyphens. The name must start with a letter and must end with a letter or a number. The AKS deployment may take 10-15 minutes

Note

When creating an AKS cluster a second resource group is automatically created to store the AKS resources. For more information see **Why are two resource groups created with AKS?**

Review your output from the **az aks create**. An example output is listed below. Note the number of parameters that were configured for the cluster vs those that were explicitly included as part of the command you executed.

```
Finished service principal creation[#####] 100.0000  
% - StartinAAD role propagation done[#####]  
] 100.0000%{  
  "aadProfile": null,  
  "addonProfiles": {  
    "omsagent": {  
      "config": {
```

```
    "logAnalyticsWorkspaceResourceID": "/subscriptions/f58e3610-6947-4b11-b5c2-f48dc74ad19b/resourcegroups/defaultresourcegroup-eus/providers/microsoft.operationalinsights/workspaces/defaultworkspace-f58e3610-6947-4b11-b5c2-f48dc74ad19b-eus"
  },
  "enabled": true,
  "identity": null
}
},
"agentPoolProfiles": [
  {
    "availabilityZones": null,
    "count": 3,
    "enableAutoScaling": null,
    "enableNodePublicIp": false,
    "maxCount": null,
    "maxPods": 110,
    "minCount": null,
    "mode": "System",
    "name": "nodepool1",
    "nodeLabels": {},
    "nodeTaints": null,
    "orchestratorVersion": "1.18.2",
    "osDiskSizeGb": 128,
    "osType": "Linux",
    "provisioningState": "Succeeded",
    "scaleSetEvictionPolicy": null,
    "scaleSetPriority": null,
    "spotMaxPrice": null,
    "tags": null,
    "type": "VirtualMachineScaleSets",
    "vmSize": "Standard_D2s_v3",
    "vnetSubnetId": null
  }
],
"apiServerAccessProfile": null,
"autoScalerProfile": null,
"diskEncryptionSetId": null,
"dnsPrefix": "gscakslab1-gscakshandsonlab-f58e36",
"enablePodSecurityPolicy": null,
"enableRbac": true,
"fqdn": "gscakslab1-gscakshandsonlab-f58e36-8b142274.hcp.eastus.azmk8s.io",
"id": "/subscriptions/f58e3610-6947-4b11-b5c2-f48dc74ad19b/resourcegroups/gscakshandsonlab/providers/Microsoft.ContainerService/managedClusters/gscakslab1",
```

```
"identity": null,
"identityProfile": null,
"kubernetesVersion": "1.18.2",
"linuxProfile": {
  "adminUsername": "azureuser",
  "ssh": {
    "publicKeys": [
      {
        "keyData": "ssh-
rsa AAAAB3NzaC1yc2EAAAADAQABAAQCMHZwbXaG0+KSGX81zK6GrhXK6rdSsTL2fSx8m1KTyDyhm5
IQld5xefwMPuMXqkRaXd3PZrWJ59djYvmQSYf8Yfq3NCIzJ06xG5K8JruVgW0dSW9dsp6TJlWaFnGgr9
1Tlds6mQsxiD6sFLRuVoruxHCuUoAfSJzFYwnypmpewK+Xakl3qE9WLQbK3JTx0ZVyx1bw+pU/MHywqrM
8SfTPGzqSjzJM6kVs7WNU0qbcxUzNbq026rsnfkyYgMORHe4Ts5o8QAjyCA75Qrz4C2uc1Dhxx7f/GrLk
CbubdVKkse1YJYRvHnyaV5JG1yZgjXkCNujJXneJ4tD+tw31zAp1"
      }
    ]
  }
},
"location": "eastus",
"maxAgentPools": 10,
"name": "gscakslab1",
"networkProfile": {
  "dnsServiceIp": "10.0.0.10",
  "dockerBridgeCidr": "172.17.0.1/16",
  "loadBalancerProfile": {
    "allocatedOutboundPorts": null,
    "effectiveOutboundIps": [
      {
        "id": "/subscriptions/f58e3610-6947-4b11-b5c2-
f48dc74ad19b/resourceGroups/MC_gscakshandsonlab_gscakslab1_eastus/providers/Micro
soft.Network/publicIPAddresses/0d6c73ad-0f3a-4cce-bcfb-8d3691aa13eb",
        "resourceGroup": "MC_gscakshandsonlab_gscakslab1_eastus"
      }
    ],
    "idleTimeoutInMinutes": null,
    "managedOutboundIps": {
      "count": 1
    },
    "outboundIpPrefixes": null,
    "outboundIps": null
  },
  "loadBalancerSku": "Standard",
  "networkMode": null,
  "networkPlugin": "kubenet",
  "networkPolicy": null,
```

```

    "outboundType": "loadBalancer",
    "podCidr": "10.244.0.0/16",
    "serviceCidr": "10.0.0.0/16"
  },
  "nodeResourceGroup": "MC_gscakshandsonlab_gscakslab1_eastus",
  "privateFqdn": null,
  "provisioningState": "Succeeded",
  "resourceGroup": "gscakshandsonlab",
  "servicePrincipalProfile": {
    "clientId": "394207d2-36ee-40a8-a821-8728614e606f",
    "secret": null
  },
  "sku": {
    "name": "Basic",
    "tier": "Free"
  },
  "tags": null,
  "type": "Microsoft.ContainerService/ManagedClusters",
  "windowsProfile": null
}

```

3. As part of your review of the output, make special note of the first line output, "Finished Service Principle creation." When you create an AKS cluster in the Azure portal or using the [az aks create](#) command, [Azure can automatically generate a service principal](#). In our example, we did not explicitly specify a service principal, so the Azure CLI creates a service principal for the AKS cluster. To successfully complete the creation of the service principal, your Azure account must have the proper rights to create a service principal. You can also manually create your own service principle using the following command.

az ad sp create-for-rbac

To create an Azure AD service principal, you must have permissions to register an application with your Azure AD tenant, and to assign the application to a role in your subscription. If you don't have the necessary permissions, you might need to ask your Azure AD or subscription administrator to assign the necessary permissions, or pre-create a service principal for you to use with the AKS cluster. [Member users](#) should have the default permissions necessary for this. The service principal will be used to access other resources. For example, if you want to connect to an Azure Container Registry (ACR) instance, which you will do in this lab, you need to delegate access to those resources to the service principal.

To delegate permissions, you can explicitly create a role assignment using the [az role assignment create](#) command. You need to grant permissions to the service principal for your AKS cluster to read and pull images. You will do this in Step 8 below.

At the end of the lab when you delete the resource group, be aware that when you delete an AKS cluster that was created by [az aks create](#), the service principal that was created automatically is not deleted. To delete the service principal, query for your cluster *servicePrincipalProfile.clientId* and then delete with [az ad app delete](#). Replace the following resource group and cluster names with your own values:

```
az ad sp delete --id $(az aks show -g myResourceGroup -n myAKSCluster --query servicePrincipalProfile.clientId -o tsv)
```

4. Check the status of your 3-node cluster from the shell using the kubectl command:

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
aks-nodepool11-62612151-vmss000000	Ready	agent	25h	v1.18.2
aks-nodepool11-62612151-vmss000001	Ready	agent	25h	v1.18.2
aks-nodepool11-62612151-vmss000002	Ready	agent	25h	v1.18.2

5. If you are asking yourself where is the YAML for the AKS provisioning? Use the following command to see the YAML. Aren't you glad you didn't have to create it? Take a browse, and in future labs we will be creating and updating the YAML directly to modify the AKS configuration. The use of provisioning based on YAML is declarative management.

```
kubectl get deployment -o yaml
```

For more information on the deployment manifest, see [AKS Deployments and YAML manifests](#).

6. So what is a Deployment? Get **Kubernetes Deployment** information using the following command. A Deployment is responsible for creating and updating instances of your application.

```
kubectl get deployments
```

Once you have a running Kubernetes cluster, you can deploy your containerized applications on top of it. To do so, you create a Kubernetes **Deployment** configuration. The Deployment instructs Kubernetes how to create and update instances of your

application. Once you've created a Deployment, the Kubernetes master schedules the application instances included in that Deployment to run on individual Nodes in the cluster. Once the application instances are created, a Kubernetes Deployment Controller continuously monitors those instances. If the Node hosting an instance goes down or is deleted, the Deployment controller replaces the instance with an instance on another Node in the cluster. **This provides a self-healing mechanism to address machine failure or maintenance.** In a pre-orchestration world, installation scripts would often be used to start applications, but they did not allow recovery from machine failure. By both creating your application instances and keeping them running across Nodes, Kubernetes Deployments provide a fundamentally different approach to application management. More on Deployments in future labs.

7. **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>
```

Important: Enter a unique ACR name. ACR name may contain alpha numeric characters only and must be between 5 and 50 characters

8. **Grant AKS-generated Service Principal access to ACR :** Authorize the AKS cluster to connect to the Azure Container Registry using the AKS generated Service Principal. Replace the variables **\$AKS_RESOURCE_GROUP**, **\$AKS_CLUSTER_NAME**, **\$ACR_RESOURCE_GROUP** with appropriate values below and run the commands.

```
# Get the id of the service principal configured for AKS
```

```
CLIENT_ID=$(az aks show --resource-group $AKS_RESOURCE_GROUP --name $AKS_CLUSTER_NAME --query "servicePrincipalProfile.clientId" --output tsv)
```

```
# Get the ACR registry resource id
```

```
ACR_ID=$(az acr show --name $ACR_NAME --resource-group $ACR_RESOURCE_GROUP --query "id" --output tsv)
```

```
# Create role assignment
```

```
az role assignment create --assignee $CLIENT_ID --role acrpull --scope $ACR_ID
```

For more information see document on how to [Authenticate with Azure Container Registry from Azure Kubernetes Service](#)

9. **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
```

Important: Enter a unique SQL server name. Since the Azure SQL Server name does not support **UPPER / Camel** casing naming conventions, use lowercase for the **SQL Server Name** field value.

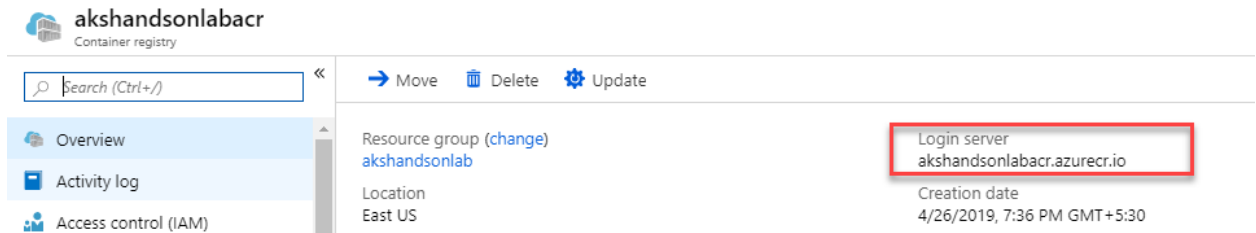
10. The following components - **Container Registry, Kubernetes Service, SQL Server** along with **SQL Database** are deployed. Access each of these components individually and make a note of the details which will be used in Exercise 1.

NAME	TYPE	LOCATION
akshandsonlab2004	Kubernetes service	East US
akshandsonlabacr	Container registry	East US
akssqlserver2604	SQL server	East US
mhcdb (akssqlserver2604/mhcdb)	SQL database	East US

11. Select the **mhcdb** SQL database and make a note of the **Server name**.

Server name
akssqlserver2604.database.windows.net

12. Navigate to the resource group, select the created container registry and make a note of the **Login server name**.

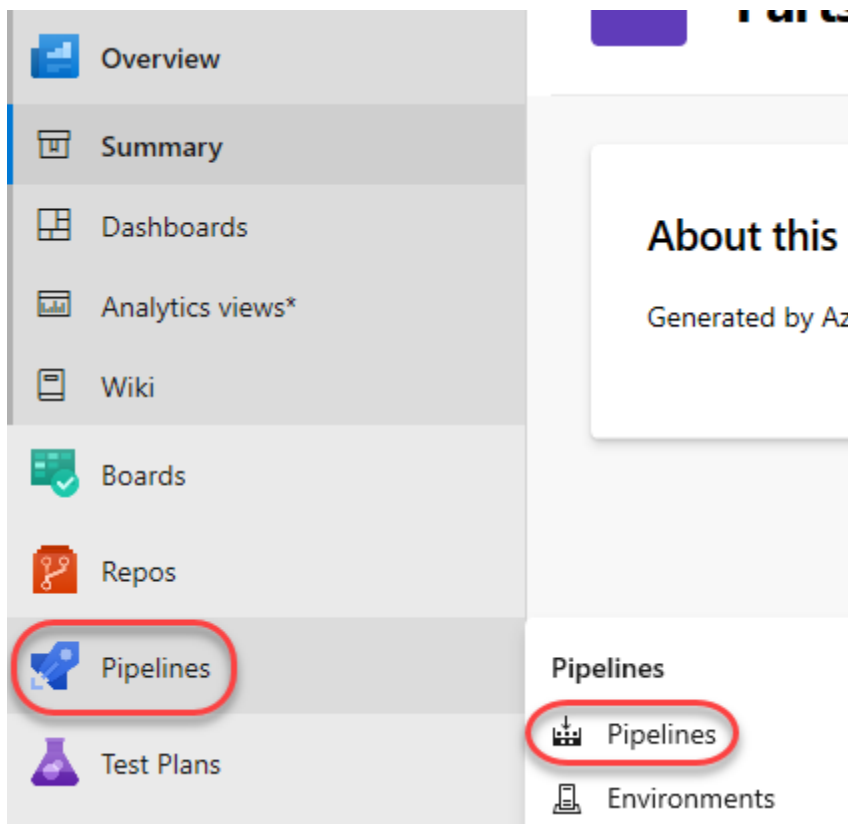


Now you have all the required infrastructure components to follow this lab. Now let's create the Build and Release Pipelines for the application deployment.

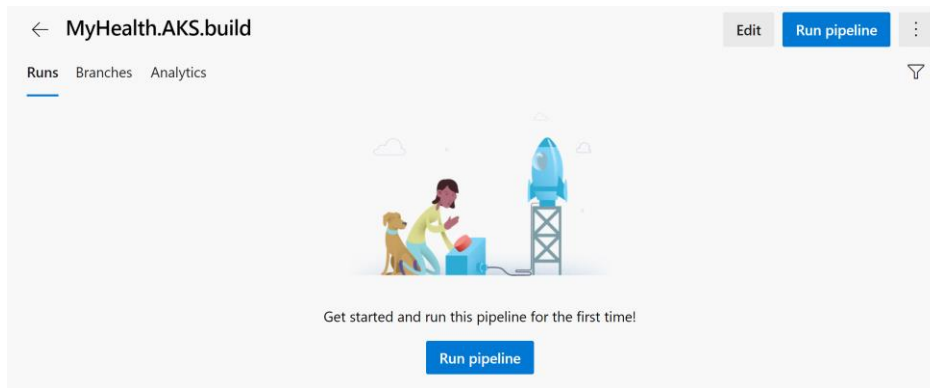
Exercise 1: Configure Build and Release pipeline

Make sure that you have created the AKS project in your Azure DevOps organization through [Azure DevOps Demo Generator](#) (as mentioned in pre-requisites). We will manually map Azure resources such as AKS and Azure Container Registry to the build and release definitions.





1. Navigate to **Pipelines** → **Pipelines**.



2. We need to create a Service Connection. Select **MyHealth.AKS.Build** pipeline and click **Edit**.

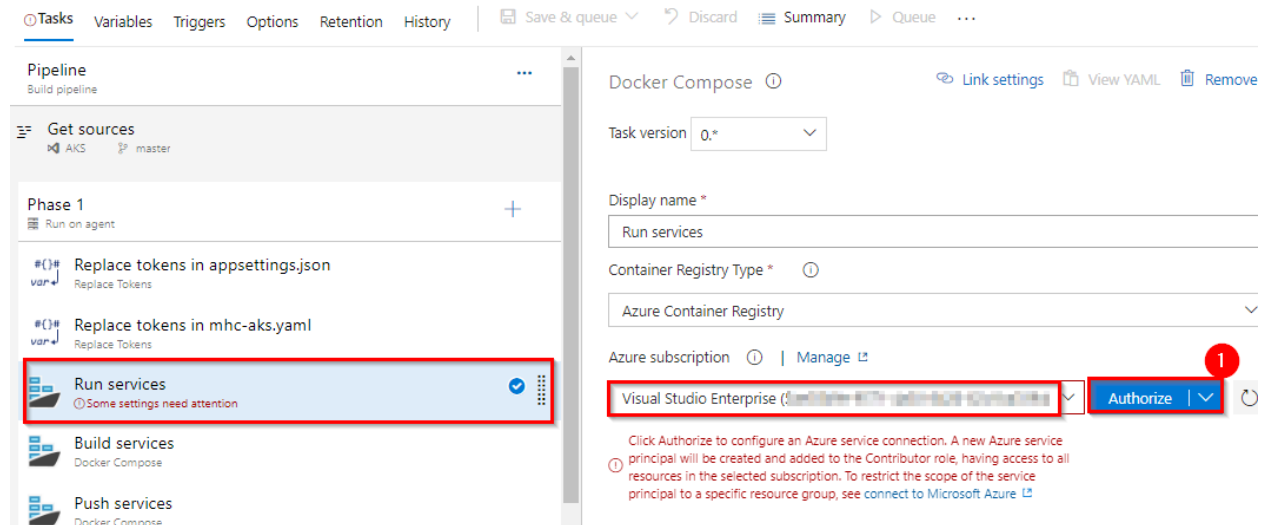


- Note the different tasks in the pipeline. The first image below summarizes each of the tasks that need to be completed. The second image illustrates the changes you need to make. In **Run services** task shown below in the second image, select your Azure subscription from **Azure subscription** dropdown. Click **Authorize**.

Tasks	Usage
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.



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 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**.

Tip: If your subscription is not listed or to specify an existing service principal, follow the [Service Principal creation](#) instructions.

- Following the successful authentication, select appropriate values from the dropdown - **Azure subscription** and **Azure Container Registry** as shown.

Repeat this for the **Build services**, **Push services** and **Lock services** tasks in the pipeline.

Pipeline
Build pipeline

Get sources
AKS master

Phase 1
Run on agent

- Replace tokens in appsettings.json
- Replace tokens in mh-aks.yaml
- Run services** (Docker Compose) - 1
- Build services (Docker Compose) - 2
- Push services (Docker Compose) - 3
- Lock services (Docker Compose) - 4

Docker Compose [Link settings](#) [View YAML](#) [Remove](#)

Task version: 0.*

Display name: Run services

Container Registry Type: Azure Container Registry

Azure subscription: Visual Studio Enterprise (5ae35b9e-9571-4a30-8426-62c1ba3394c4)

Azure Container Registry: [Redacted]

Docker Compose File: docker-compose.ci.build.yml

5. Click on **Pipeline**, and then click on the **Variables** tab.

MyHealth.AKS.build

Tasks **Variables** Triggers Options Retention History

Pipeline
Build pipeline

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

MyHealth.AKS.build

Tasks **Variables** Triggers Options Retention History [Save & queue](#) [Discard](#) [Summary](#) [Queue](#) [...](#)

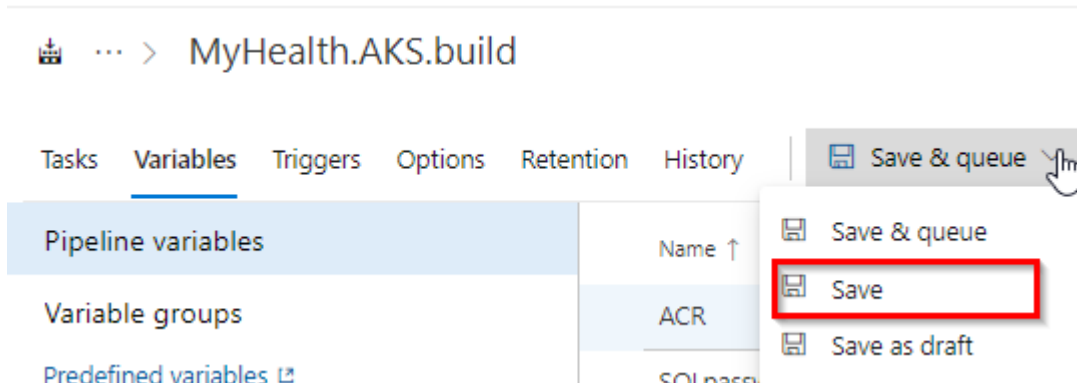
Pipeline variables

Variable groups

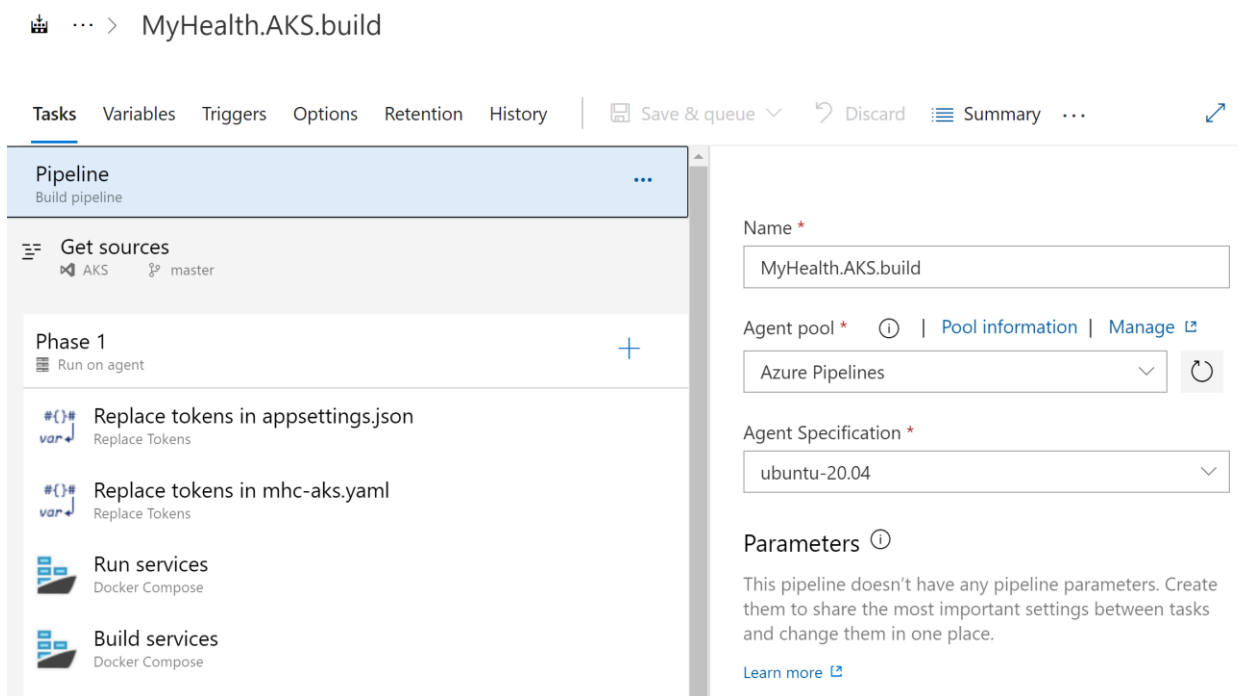
[Predefined variables](#)

Name ↑	Value
ACR	azurecr.io
SQLpassword	P2ssw0rd1234
SQLserver	database.windows.net
SQLuser	sqladmin
system.collectionId	98b99345-6453-4de5-9e74-5cdcd3bd22e
system.debug	false
system.definitionId	132

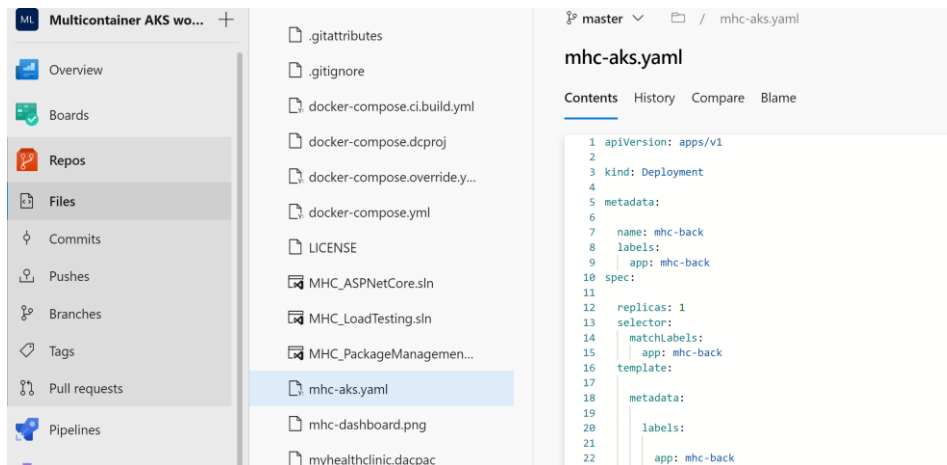
6. **Save** the changes.



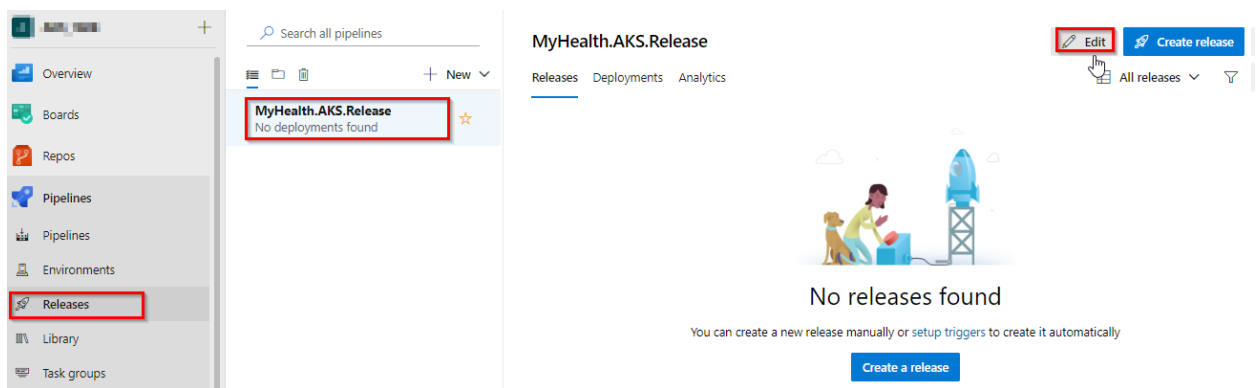
7. Select Pipeline and make sure that **Azure Pipelines** is selected for the **Agent pool**, and **ubuntu-20.04** is selected for **Agent Specification** as shown below.



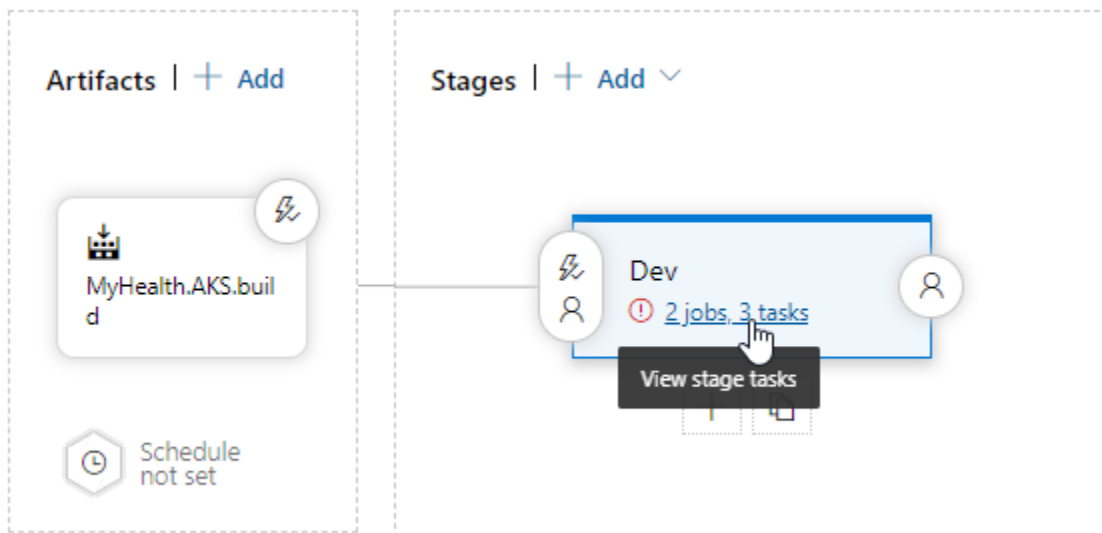
8. **Save** the changes.
9. If you'd like to see the YAML that is auto-created for you then select **Repos** in the left-hand menu and click on the **mhc-aks.yaml** file. **DO NOT MAKE ANY CHANGES.**



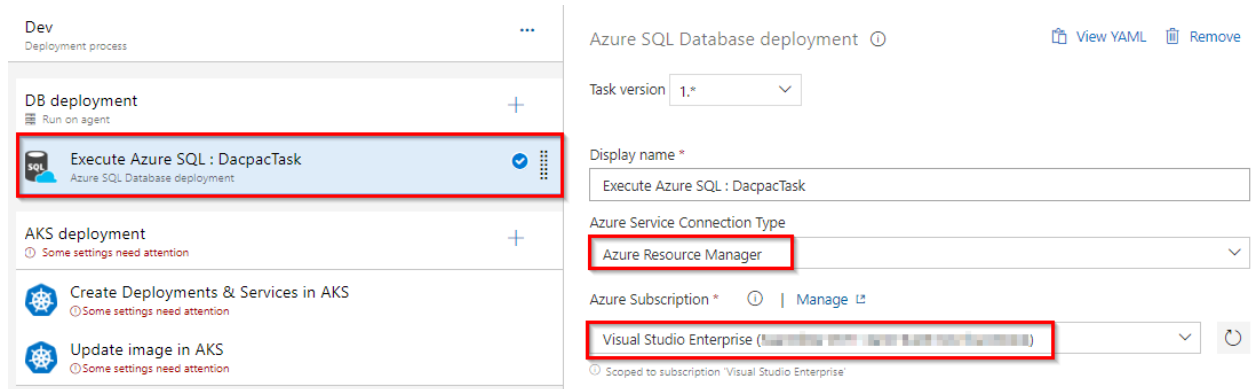
10. Navigate to **Pipelines | Releases**. Select **MyHealth.AKS.Release** pipeline and click **Edit**.



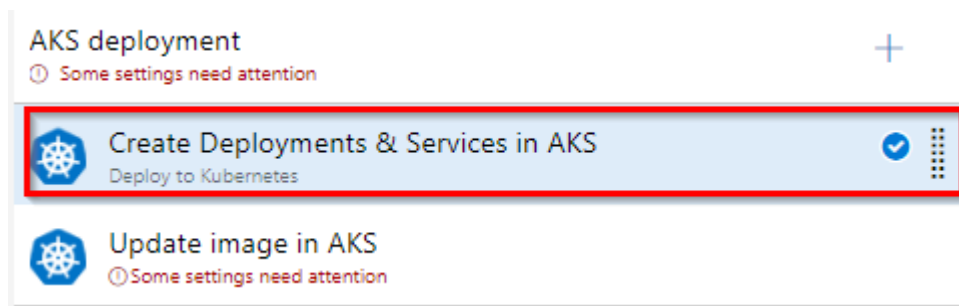
11. Select **Dev stage** and click **View stage tasks** to view the pipeline tasks.



12. In the **Dev** environment, select **Execute Azure SQL: DacpacTask** under the **DB deployment** heading, select **Azure Resource Manager** from the drop down for **Azure Service Connection Type** (it may already be selected), and update the **Azure Subscription** value from the dropdown for **Execute Azure SQL: DacpacTask** task.



13. 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.

Deploy to Kubernetes ①

 View YAML  Remove

Task version 1.* ▾

Display name *

Create Deployments & Services in AKS

Kubernetes Cluster ^

Service connection type * ①

Azure Resource Manager ▾

Azure subscription * ① | [Manage](#)

Visual Studio Enterprise () ▾ ↻

① Scoped to subscription 'Visual Studio Enterprise'

Resource group * ①

akshandsonlab ▾ ↻

Kubernetes cluster * ①

) ▾ ↻

☐ Use cluster admin credentials ①

Namespace ①

Commands ▾

Secrets ^

Type of secret * ①

dockerRegistry ▾

Container registry type * ①

Azure Container Registry ▾

Azure subscription ① | [Manage](#)

Visual Studio Enterprise () ▾ ↻

① Scoped to subscription 'Visual Studio Enterprise'

Azure container registry ①

\$(ACR) ▾ ↻

Secret name ①

mysecretkey

☒ Force update secret ①

ConfigMaps ▾

Advanced ▾

- **Create Deployments & Services in AKS** will create the deployments and services in AKS as per the configuration specified in **mhc-aks.yaml** file. The Pod, for the first time will pull up the latest docker image.
- **Update image in AKS** will pull up the appropriate image corresponding to the BuildID from the repository specified, and deploys the docker image to the **mhc-front pod** running in AKS.
- A secret called **mysecretkey** is created in AKS cluster through Azure DevOps by using command **kubecti create secret** in the background. This secret will be used for authorization while pulling myhealth.web image from the Azure Container Registry.

14. Make sure you check your Agent Job configuration as shown below. You will need **windows-2019** for your **Agent Specification**. Why do you need Windows VMs, you used Ubuntu for the Build?

All pipelines > MyHealth.AKS.Release

Save Create release ...

Pipeline **Tasks** ▾ Variables Retention Options History

Dev
Deployment process

DB deployment
Run on agent

Execute Azure SQL : DacpacTask
Azure SQL Database deployment

AKS deployment
Run on agent

Create Deployments & Services in AKS
Kubecti

Update image in AKS
Kubecti

Agent job ⓘ Remove

Display name *
DB deployment

Agent selection ^

Agent pool ⓘ | [Pool information](#) | [Manage](#)

Azure Pipelines

Agent Specification *
windows-2019

Demands ⓘ

15. 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.

Note: The **Database Name** is set to **mhcdb** and the **Server Admin Login** is **sqladmin** and **Password** is **P2ssw0rd1234**. If you have entered different details while creating Azure SQL server, update the values accordingly

All pipelines > MyHealth.AKS.Release

Pipeline Tasks Variables Retention Options History

Pipeline variables

Variable groups

Predefined variables

Filter by keywords Scope

Name	Value
ACR	myhealth.azurecr.io
DatabaseName	mhcdb
SQLpassword	P2ssw0rd1234
SQLserver	database.windows.net
SQLuser	sqladmin

Exercise 2: Trigger a Build and deploy application

In this exercise, let us trigger a build manually and upon completion, an automatic deployment of the application will be triggered. Our application is designed to be deployed in the pod with the **load balancer** in the front-end and **Redis cache** in the back-end.

1. Select **MyHealth.AKS.build** pipeline. Click on **Run pipeline**

MyHealth.AKS.build

Runs Branches Analytics

Get started and run this pipeline for the first time!

Run pipeline

2. Once the build process starts, select the build job to see the build in progress.

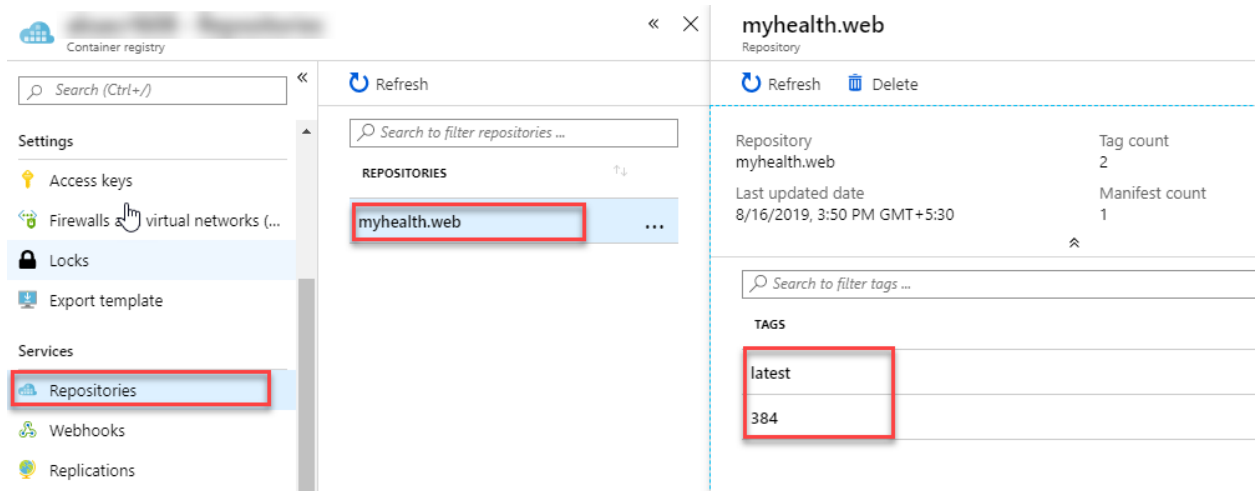
MyHealth.AKS.build

Runs Branches Analytics

Get started and run this pipeline for the first time!

Run pipeline

- 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**.



- Switch back to the Azure DevOps portal. Select the **Releases** tab in the **Pipelines** section and double-click on the latest release. Select **In progress** link to see the live logs and release summary.

↑ MyHealth.AKS.Release > Release-3 ↓

Pipeline Variables History | + Deploy ▼ | ⏹ Cancel | ↻ Refresh | ✎ Edit ▼ | ...

Release

Continuous deployment
for Steve Caravajal
6/16/2020, 5:47 PM

Artifacts

MyHealth.AKS.build
20200616.5
🔗 master

Stages

Dev
 In progress for 1m

Job 1/2

3/3 tasks

Execute Azure SQL :
🕒 01:39

MyHealth.AKS.Release > Release-3 > Dev ✓ Succeeded

← Pipeline Tasks Variables **Logs** Tests | [Deploy](#) [Cancel](#) [Refresh](#) [Download all logs](#) ...

Deployment process
Succeeded

- ✓ DB deployment
Succeeded
- ✓ AKS deployment
Succeeded

DB deployment Started: 6/16/2020, 5:48:07 PM

Pool: [Azure Pipelines](#) · Agent: Hosted Agent ... 2m 32s

✓ Initialize job · succeeded	2s
✓ Download artifact - MyHealth.AKS.build - depl... · succeeded	2s
✓ Execute Azure SQL : DacpacTask · succeeded	2m 26s
✓ Finalize Job · succeeded	<1s

- Once the release is complete, launch the [Azure Cloud Shell](#) and run the commands below to see the pods running in AKS:

```
az aks get-credentials --resource-group yourResourceGroup --name yourAKSname
```

```
kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
mhc-back-6f77bc9679-qx67g	1/1	Running	0	91m
mhc-front-547fb96dbb-bskxf	1/1	Running	0	91m

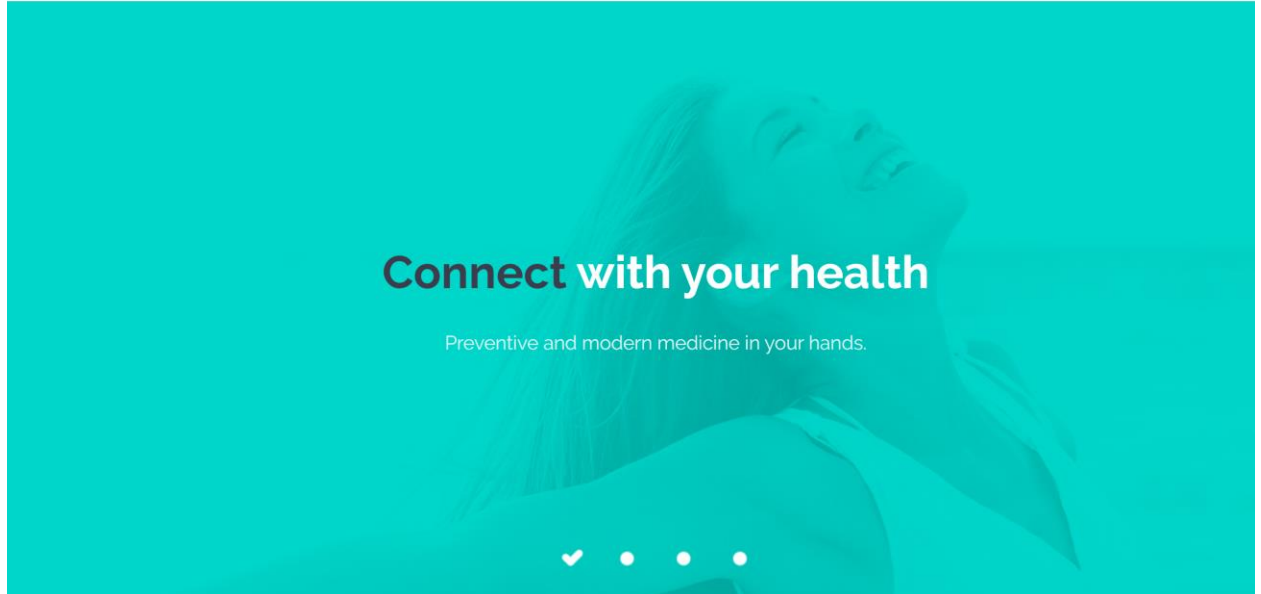
The deployed web application is running in the displayed pods.

- To access the application, run the below command. If you see that **External-IP** is pending, wait for sometime until an IP is assigned.

```
kubectl get service mhc-front --watch
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
mhc-front	LoadBalancer	10.0.71.96	52.191.228.135	80:30682/TCP	94m

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



After the hands-on lab

In this exercise, you will de-provision any Azure resources created in support of this lab.

1. Delete the Resource Groups in which you placed all your Azure resources.
 - From the Portal, navigate to the blade of your Resource Group and then select Delete in the command bar at the top.
 - Confirm the deletion by re-typing the resource group name and selecting Delete.
2. Delete the Service Principal created in Step 3, using the directions specified for deleting the Service Principal.

Challenge 1

Using Azure DevOps, provision the AKS cluster using ARM templates. Make sure to de-provision your resources once complete.

