**Q3 - SCENARIO**

A Toy Retail company ToyTrex has it retail application deployed as 3-tier application - Web App(UI), Web API(middle layer) and Database as Azure SQL.

The user load started increasing multiple fold every month and complex programs getting implemented, the application started performing poorly.

As a result, company decided to re-architect the middle layer as microservices using Azure Kubernetes Services.

The new architecture has below design decisions.

1) The middle layer should be implemented as Microservices using Azure AKS

2) The middle layer API should be deployed as containerized application images

3) The container images will use Azure Container Repository (ACR) as the private image repository

4) The CI/CD pipelines for microservices should be implemented using Azure DevOps services.

5) The Azure DevOps should be able to access ACR and download the container images for microservices deployment

6) The image should be deployed as templates such as <image\_name>:<build\_id>

Explain the DevOps configuration and steps in detail for above requirements

Implementation

Microservice architecture with Azure AKS

1. First create Azure Kubernetes cluster at Azure. Create resource group and select subscription to go ahead with configuring AKS cluster.

2. Create a namespace on the for your app

2. To deploy azure microservice we need to Setup service connection to

this cluster in azure devops

3. Go into the microservice project solution in Visual studio and add docker file to solution.

4. Create a ACR on Azure portal using the information on resource group and subscription information.

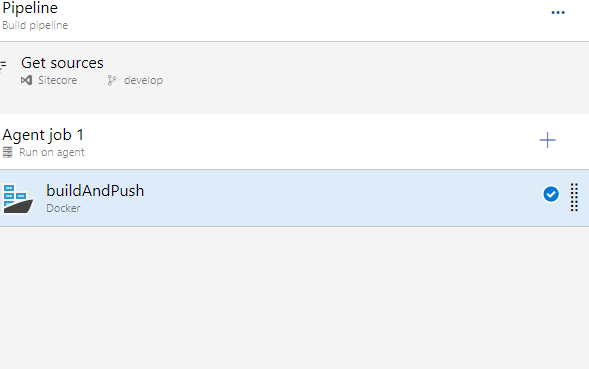
5. Also create service connection for ACR.

Build :

1. Go to Azure DevOps create a new build pipeline.

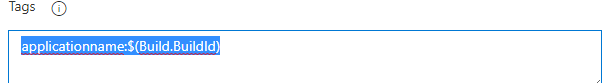
2. In the Build definition we have to create new docker image daily when ever commit comes into branch.

3. Enable continuous integration.



4. Add the new service connection.

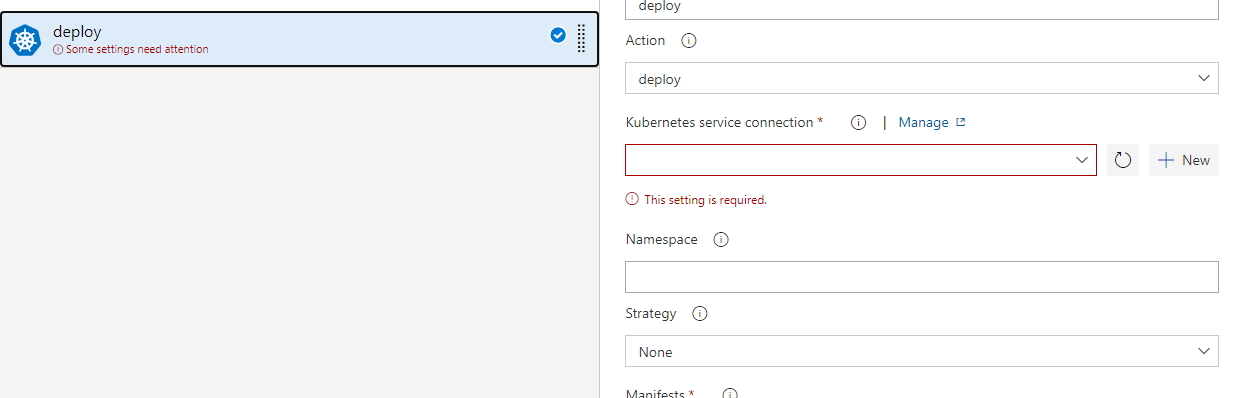
5. use the below to tag.



Release :

1. Create new release pipeline.

2. Add a task deploy to Kubernetes.



3. select the service connection, namespace and manifest the image name and the image pull secret