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.

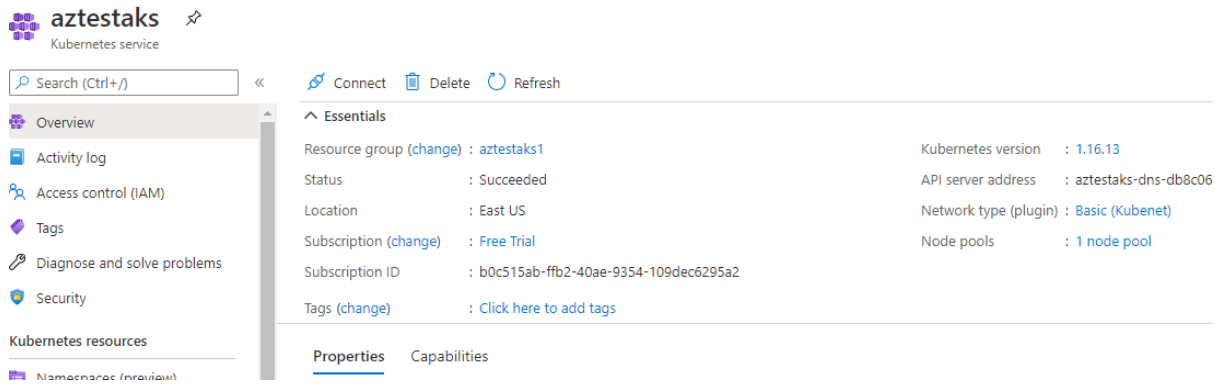
5) The Azure DevOps should be able to access ACR and download the container images for microservices deployment

6) The image should be deployed as templates such as <image_name>:<build_id>

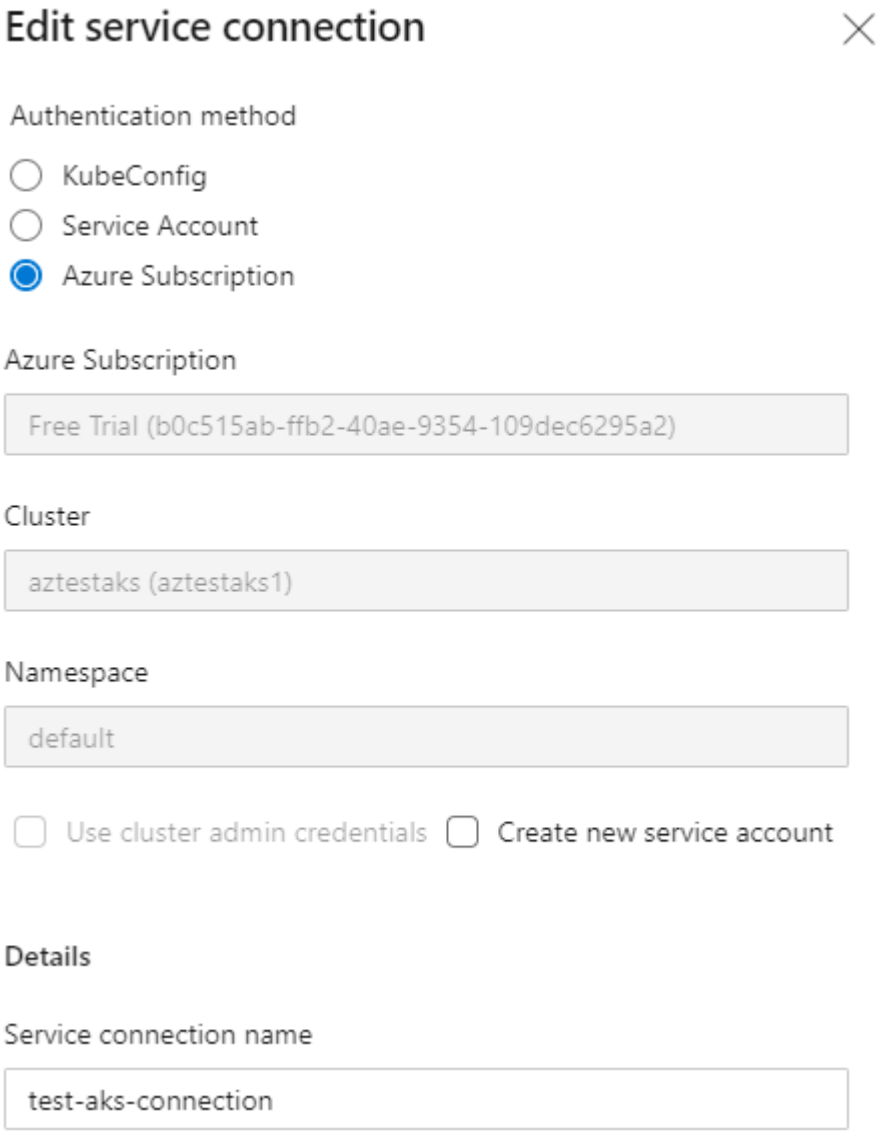
Explain the DevOps configuration and steps in detail for above requirements

Solution3:

1) To implement Microservice architecture with Azure AKS, first create Azure Kubernetes cluster at Azure. Create resource group and select subscription to go ahead with configuring AKS cluster.

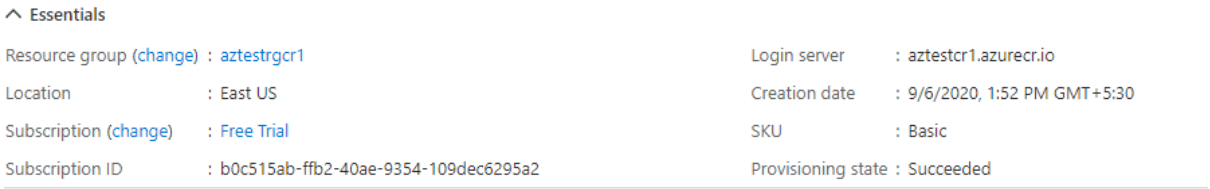


This cluster contains 2 nodes in node pool.
To use the AKS cluster to deploy azure microservice we need to Setup service connection to this cluster in azure devops

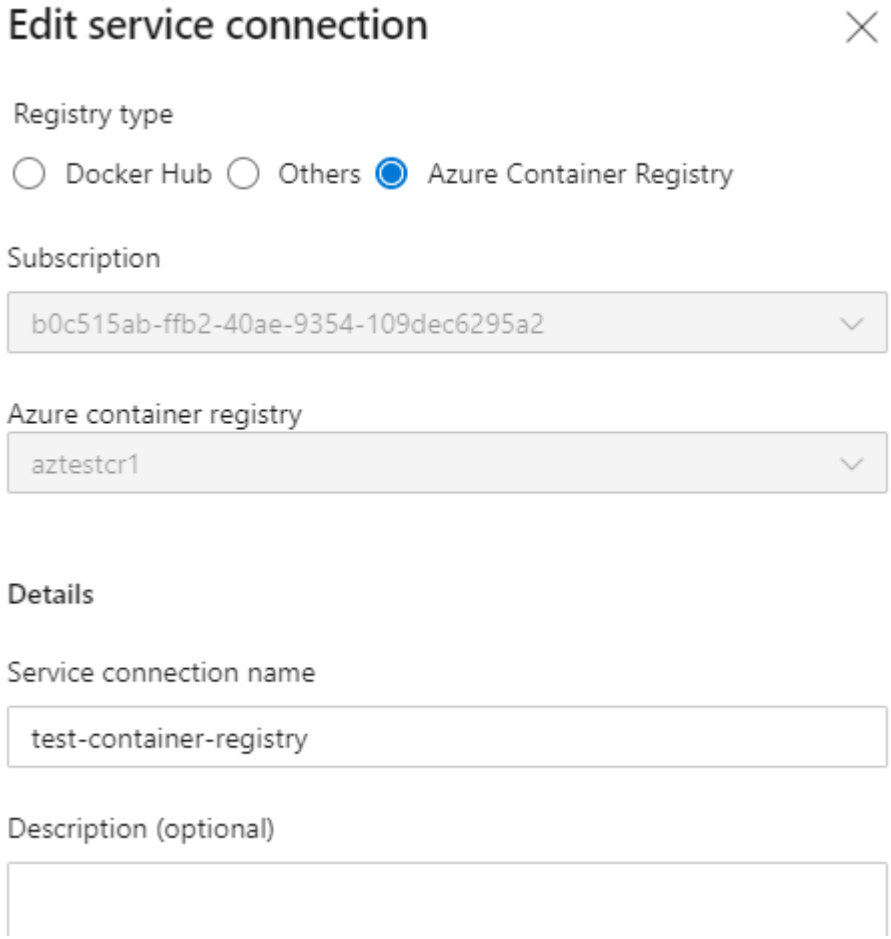


2) Check-in the Docker file to create containerized image of microservice to source code repository along with source code. This Dockerfile will be used in Azure devops pipeline to create image of microservice.

3) Create Azure container Registry in Microsoft azure portal. Give resource group and subscription information along with region information to create container registry.



Create service connection of created ACR with Azure Devops settings.



4) Start creating new starter pipeline in Azure Devops using applicable source code repository. (one can use inbuilt docker or Kubernetes pipelines as well with customized parameters)

5) Create yaml configuration for pipeline for our requirement.

```yaml
trigger:
- master

resources:
- repo: self

variables:
  tag: '$(Build.BuildId)'

stages:
- stage: Build
  displayName: Build image
  jobs:
  - job: Build
    displayName: Build
    pool:
      vmImage: 'ubuntu-latest'
```

Image should be append using build ID as tag (image:buildID)

```
    steps:
    - task: Docker@2
      displayName: Build and Push an image
      inputs:
        containerRegistry: 'azure-container-registry'
        repository: 'aztestcr1.azurecr.io/toyTrex'
        command: 'buildAndPush'
        Dockerfile: '**/Dockerfile'
        tags: '$(tag)'
    - task: CopyFiles@2
      inputs:
        SourceFolder: '$(System.DefaultWorkingDirectory)'
        Contents: '**/*.yaml'
        TargetFolder: '$(Build.ArtifactStagingDirectory)'
    - task: PublishBuildArtifacts@1
      inputs:
        PathtoPublish: '$(Build.ArtifactStagingDirectory)'
        ArtifactName: 'manifests'
        publishLocation: 'Container'

- stage: Deploy
  displayName: Deploy image
  jobs:
  - job: Deploy
    displayName: Deploy
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - task: DownloadPipelineArtifact@2
      inputs:
        buildType: 'current'
        artifactName: 'manifests'
        itemPattern: '**/*.yaml'
        targetPath: '$(System.ArtifactsDirectory)'
    - task: KubernetesManifest@0
      inputs:
        action: 'deploy'
        kubernetesServiceConnection: 'test-aks-connection'
        namespace: 'default'
        manifests: '$(System.ArtifactsDirectory)/configuration/kubernetes/depl
oyment.yaml'
        containers: 'aztestcr1.azurecr.io/toyTrex:$(tag)'
```

Building and Pushing docker image to ACR with Tag defined under variables

Deploying the image to Kubernetes cluster created in AKS