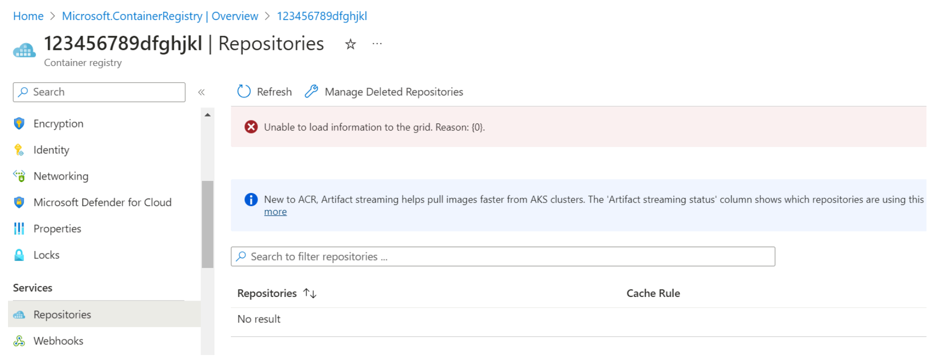
Chat-bot Monolithic deployment to kubernetes :-

1. Container create :-

When u open to see the docker images u will see this error

Unable to load information to the grid. Reason: {0}.



Home > Microsoft.ContainerRegistry | Overview > 123456789dfghjkl

123456789dfghjkl | Repositories

Container registry

Unable to load information to the grid. Reason: {0}.

New to ACR, Artifact streaming helps pull images faster from AKS clusters. The 'Artifact streaming status' column shows which repositories are using this more

Repositories ↑↓	Cache Rule
No result	

Search
Encryption
Identity
Networking
Microsoft Defender for Cloud
Properties
Locks
Services
Repositories
Webhooks
Refresh
Manage Deleted Repositories
Search to filter repositories ...

To resolve this u do this and it will get resloved

Go to IAM —> give him the owner permission

Go to Access key—> give Admin user ✅

2. Service connection :-

For that container create a service connection

Go to Azure devops portal and do like this

Project settings —> Service connection —> New Service connection —>

Select docker Docker Registry —-> Azure Container Registry —-> save.

3. Build Docker Images :-

Now with the code build a docker image and push into your docker container registry u can refer to the yaml file u will understand.

4. Kubernetes services :-

    Now select 1st option Create Kubernetes cluster and fill the necessary thing it is asking and create

5. Service connection :-

    Now that Kubernetes cluster create a service connection

    Go to Azure devops portal and do like this

    Project settings —> Service connection —> New Service connection —> Select Kubernetes —-> fill necessary things and create and save.

Azure-pipeline.yaml :-

```yaml
trigger:
- none

resources:
- repo: self

variables:
  containerRegistry: 'sample container 8899.azure.io'
  dockerRegistryServiceConnection: 'af96df78-8263-4dba-bb43-5ad37206afce'
  baseImageName: 'chatbot-base'
  appImageName: 'chatbot'
  aksClusterName: 'samplecluster89'
  aksSubscriptionEndpoint: 'Azure for Students'
  tag: 'latest'


stages:
- stage: BuildAndPushBaseImage
  displayName: 'Build and Push Base Image'
  jobs:
  - job: BuildBaseImage
    displayName: 'Build Base Image'
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - task: Docker@2
      displayName: 'Build and push base image to container registry'
      inputs:
        command: 'buildAndPush'
        repository: '$(baseImageName)'
        dockerfile: '$(Build.SourcesDirectory)/base.Dockerfile'
        containerRegistry: '$(dockerRegistryServiceConnection)'
        tags: 'latest'

- stage: BuildAndPushAppImage
  displayName: 'Build and Push App Image'
  jobs:
  - job: BuildAppImage
    displayName: 'Build Application Image'
    pool:
      vmImage: 'ubuntu-latest'
    steps:
    - task: Docker@2
      displayName: 'Build and push application image to container
registry'
      inputs:
        command: 'buildAndPush'
        repository: '$(appImageName)'
        dockerfile: '$(Build.SourcesDirectory)/chatbot.Dockerfile'
        containerRegistry: '$(dockerRegistryServiceConnection)'
        tags: 'latest'

- stage: Deploy
  displayName: 'Deploy stage'
  dependsOn:
  - BuildAndPushBaseImage
  - BuildAndPushAppImage
  jobs:
  - deployment: Deploy
    displayName: 'Deploy'
    pool:
      vmImage: 'ubuntu-latest'
    environment: 'myrepo.kube-node-lease' # Ensure this matches the environment name set up in Azure DevOps
    strategy:
      runOnce:
        deploy:
          steps:
          - checkout: self # This ensures that the repository contents are available
          - task: KubernetesManifest@0
            displayName: 'Deploy to Kubernetes cluster'
            inputs:
              action: 'deploy'
              kubernetesServiceConnection: 'k8sserviceconnection'
              namespace: 'kube-node-lease'
              manifests: |
                $(Build.SourcesDirectory)/manifests/deployment.yml
                $(Build.SourcesDirectory)/service.yml
              containers: 'samplecontainer8899.azurecr.io/chatbot:latest'
              imagePullSecrets: 'imagepullsecret'
              rolloutStatusTimeout: '300'
```

Important points :-

1. The Project overview is it is a java application basically it is build on Tomcat server inside that Tomcat Server the chatbot application is build
2. Here we need to build 2 docker images 1st it create a base.Docker image and after that by using 1st image it will create 2nd image
3. By using 2nd image only we will deploy this image into kubernetes cluster
4. We need to deploy only the 2nd image that will create the chatbot application.