Creating Different Microservices in Azure Pipelines

This repository contains the setup instructions and implementation details for configuring Azure DevOps for continuous integration (CI) pipelines for a project with multiple microservices. The project utilizes Azure DevOps for version control, CI pipeline configuration, and deployment.

Repository Setup

Step 1: Import Code into Azure Repo

1. Import the project code into Azure Repos.
2. Ensure proper folder structure and organization within the repository.

   `https://github.com/dockersamples/example-voting-app.git`

Step 2: Setup Self-Hosted Agent

1. Set up a self-hosted agent on a virtual machine in Azure.
2. Ensure the agent has necessary permissions and dependencies installed for building and deploying the project. ### Step 3: Create Virtual Machine
3. Create a virtual machine in Azure to host the self-hosted agent.
4. Configure the virtual machine specifications according to the project requirements. ### Step 4: Configure Container Registry
5. Set up a container registry in Azure to run the microservices.
6. Ensure proper permissions and access control for managing container images. ## Microservices Setup The project consists of three microservices:
7. **Vote Service**: Yaml File

```
# Docker
# Build and push an image to Azure Container Registry
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
  paths:
    include:
      - vote/*



resources:
- repo: self
```

```
variables:
  # Container registry service connection established during pipeline
creation
  dockerRegistryServiceConnection: 'c601d70c-7fce-4e41-8c1e-f57ddce977b4'
  imageRepository: 'resultapp'
  containerRegistry: 'dipanjancicd.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/result/Dockerfile'
  tag: '$(Build.BuildId)'

  pool:
    name:'azureagent'


stages:
- stage: Build
  displayName: Build
  jobs:
  - job: Build
    displayName: Build

    steps:
    - task: Docker@2
      displayName: Build and push an image to container registry
      inputs:
        containerRegistry: '$(dockerRegistryServiceConnection)'
        repository: '$(imageRepository)'
        command: 'build'
        Dockerfile: 'vote/Dockerfile'
        tags: '$(tag)'
- stage: push
  displayName: push
  jobs:
  - job: push
    displayName: push

    steps:
    - task: Docker@2
      displayName: push an image to container registry
      inputs:
        containerRegistry: '$(dockerRegistryServiceConnection)'
        repository: '$(imageRepository)'
        command: 'push'
        tags: '$(tag)'
```

2.Result Service: Yaml File

```
# Docker
# Build and push an image to Azure Container Registry
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker
```

```
trigger:
  paths:
    include:
      - result/*


resources:
- repo: self

variables:
  # Container registry service connection established during pipeline
creation
  dockerRegistryServiceConnection: 'c601d70c-7fce-4e41-8c1e-f57ddce977b4'
  imageRepository: 'resultapp'
  containerRegistry: 'dipanjancicd.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/result/Dockerfile'
  tag: '$(Build.BuildId)'

  pool:
    name:'azureagent'


stages:
- stage: Build
  displayName: Build
  jobs:
  - job: Build
    displayName: Build

    steps:
    - task: Docker@2
      displayName: Build and push an image to container registry
      inputs:
        containerRegistry: '$(dockerRegistryServiceConnection)'
        repository: '$(imageRepository)'
        command: 'build'
        Dockerfile: 'result/Dockerfile'
        tags: '$(tag)'
- stage: push
  displayName: push
  jobs:
  - job: push
    displayName: push

    steps:
    - task: Docker@2
      displayName: push an image to container registry
      inputs:
```

```
        containerRegistry: '$(dockerRegistryServiceConnection)'
        repository: '$(imageRepository)'
        command: 'push'
        tags: '$(tag)'
```

3.Worker Service: Yaml File

```
# Docker
# Build and push an image to Azure Container Registry
# https://docs.microsoft.com/azure/devops/pipelines/languages/docker

trigger:
  paths:
    include:
      - worker/*


resources:
- repo: self

variables:
  # Container registry service connection established during pipeline
creation
  dockerRegistryServiceConnection: 'c601d70c-7fce-4e41-8c1e-f57ddce977b4'
  imageRepository: 'resultapp'
  containerRegistry: 'dipanjancicd.azurecr.io'
  dockerfilePath: '$(Build.SourcesDirectory)/result/Dockerfile'
  tag: '$(Build.BuildId)'

  pool:
    name:'azureagent'


stages:
- stage: Build
  displayName: Build
  jobs:
  - job: Build
    displayName: Build

    steps:
    - task: Docker@2
      displayName: Build and push an image to container registry
      inputs:
        containerRegistry: '$(dockerRegistryServiceConnection)'
        repository: '$(imageRepository)'
        command: 'build'
        Dockerfile: 'worker/Dockerfile'
        tags: '$(tag)'
- stage: push
```

```
displayName: push
jobs:
- job: push
  displayName: push

  steps:
  - task: Docker@2
    displayName: push an image to container registry
    inputs:
      containerRegistry: '$(dockerRegistryServiceConnection)'
      repository: '$(imageRepository)'
      command: 'push'
      tags: '$(tag)'
```

CI Pipeline Configuration

Step 1: YAML File Configuration

1. Create YAML files for configuring CI pipelines for each microservice.
2. Define stages, jobs, and steps within each YAML file according to the build and deployment requirements of the microservice.

Step 2: Pipeline Triggers

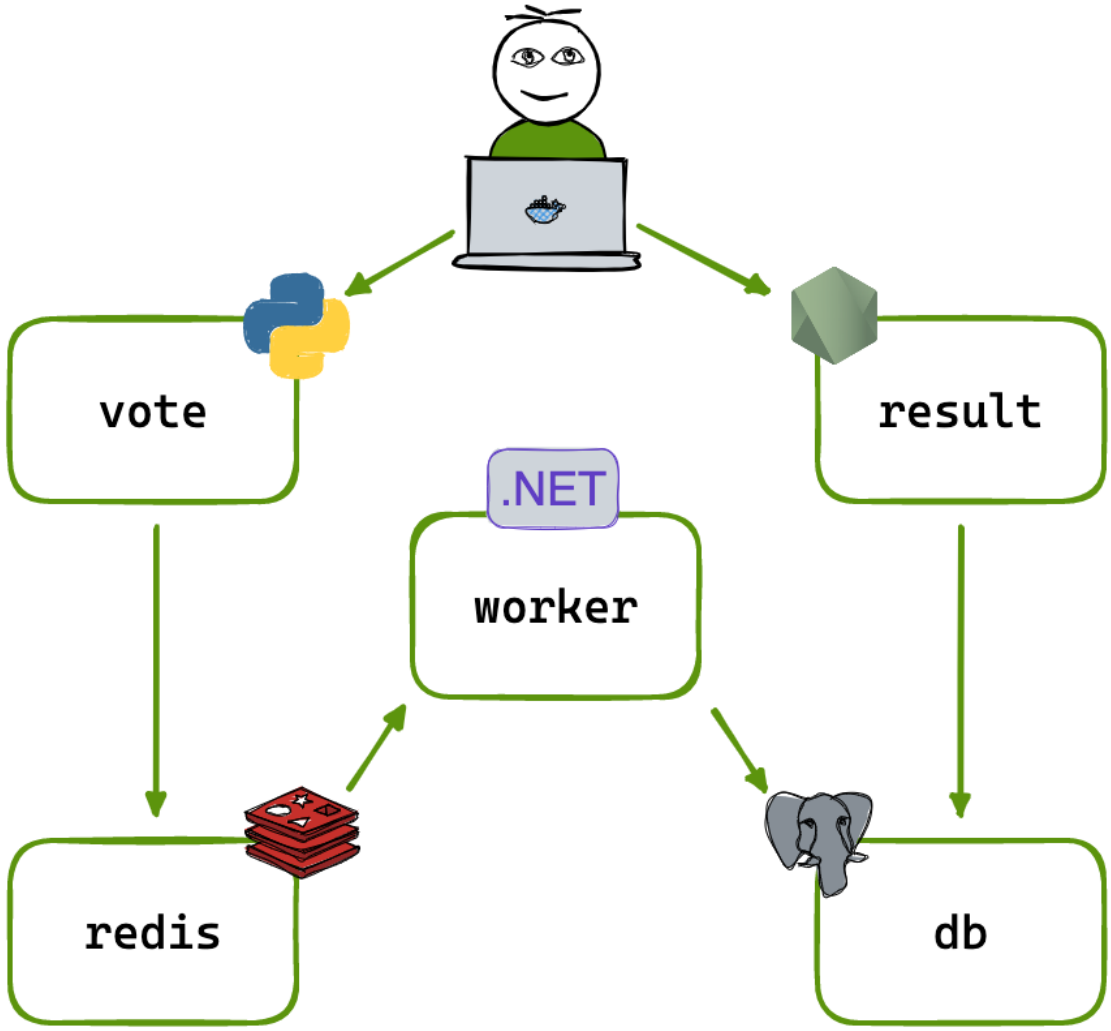
1. Configure triggers for CI pipelines to run on code commits or pull requests.
2. Define branch filters and paths as necessary to trigger the pipelines for specific changes.

Step 3: Environment Configuration

1. Define environments in Azure DevOps to facilitate deployment.
2. Configure environment-specific variables, settings, and secrets for each environment.

Usage

1. Clone this repository to your local machine.
2. Follow the setup instructions mentioned above.
3. Refer to individual YAML files for detailed configuration of CI pipelines for each microservice.



Alt Text