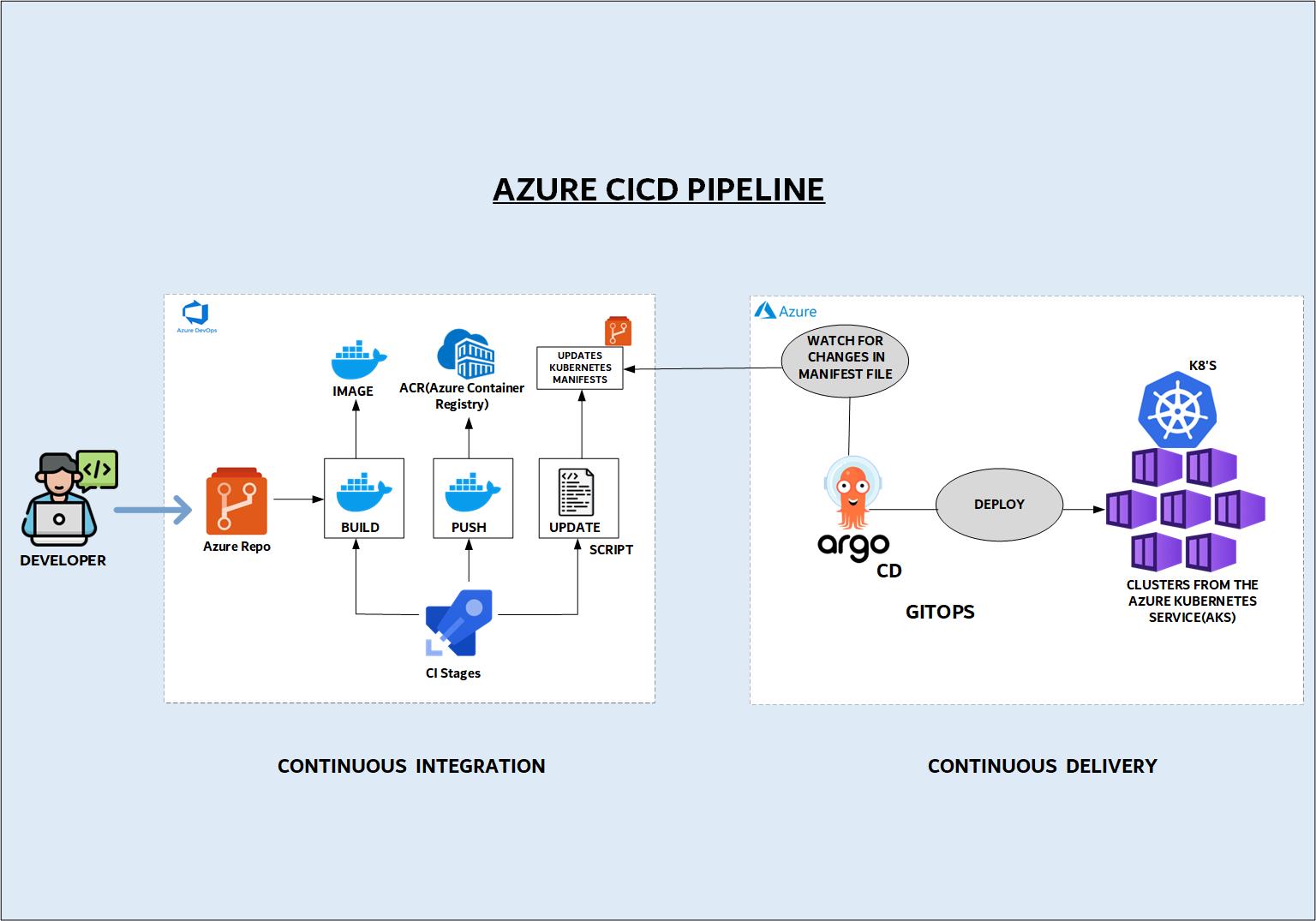
**Detailed Step-by-Step Documentation**



**Prerequisites**

1. Azure account with access to DevOps, ACR, and AKS.
2. Source code repository (e.g., Docker’s voting app sample).

**Step 1: Set Up Azure Resources**

1. **Create Azure Container Registry (ACR)**:
   * Navigate to Azure Portal → Create Resource → Container Registry.
   * Name: azure-cicd-acr, Resource Group: azure-cicd-rg.
2. **Create AKS Cluster**:
   * Azure Portal → Create Resource → Kubernetes Service (AKS).
   * Configure node pools, auto-scaling, and networking.

**Step 2: Import Repository to Azure DevOps**

1. **Import GitHub Repo**:
   * In Azure DevOps, create a project → Repos → Import Repository.
   * Use the Docker voting app sample: https://github.com/dockersamples/example-voting-app.
2. **Set Default Branch**:
   * Ensure main is the default branch.

**Step 3: Configure CI Pipelines**

1. **Create Pipeline for Each Microservice**:
   * Navigate to Pipelines → New Pipeline → Azure Repos Git.
   * Use YAML templates for **build**, **push**, and **update** stages.
2. **Example YAML Snippet (voting-app CI)**:

yaml

Copy

trigger:

paths:

include:

- voting-app/\*

stages:

- stage: Build

jobs:

- job: Build\_Image

pool: azure-agent # Self-hosted agent

steps:

- task: Docker@2

inputs:

command: build

repository: voting-app

dockerfile: voting-app/Dockerfile

tags: $(Build.BuildId)

- stage: Push

jobs:

- job: Push\_Image

steps:

- task: Docker@2

inputs:

command: push

repository: voting-app

- stage: Update\_Manifest

jobs:

- job: Update\_K8s\_Manifest

steps:

- script: ./scripts/update-k8s-manifest.sh voting-app $(Build.BuildId)

1. **Self-Hosted Agent Setup**:
   * Deploy a VM in Azure → Install Docker and Azure Pipelines Agent.
   * Register the agent in Azure DevOps → Project Settings → Agent Pools.

**Step 4: Configure GitOps with Argo CD**

1. **Install Argo CD on AKS**:

kubectl create namespace argocd

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

1. **Expose Argo CD UI**:

kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'

1. **Connect Argo CD to Azure Repos**:
   * In Argo CD UI → Settings → Repositories → Add HTTPS repo URL.
   * Use Azure Repos URL with a **Personal Access Token (PAT)** for authentication.
2. **Create Argo CD Application**:
   * Sync Policy: **Automatic** (every 10 seconds).
   * Path: /k8s-specs (folder with Kubernetes manifests).
   * Destination: AKS cluster and default namespace.

**Step 5: Automate Kubernetes Manifest Updates**

1. **Shell Script (**update-k8s-manifest.sh**)**:
   * Updates image tags in Kubernetes manifests after CI builds.

#!/bin/bash

set -x

REPO\_URL="https://<PAT>@dev.azure.com/your-org/your-project/\_git/repo"

git clone $REPO\_URL /tmp/repo

sed -i "s|image:.\*|image: azure-cicd-acr.azurecr.io/$1:$2|g" /tmp/repo/k8s-specs/$1-deployment.yaml

cd /tmp/repo && git commit -am "Update $1 image to $2" && git push

**Step 6: Validate End-to-End Flow**

1. **Test CI/CD**:
   * Modify code in voting-app → Commit to Azure Repos.
   * CI Pipeline triggers → Builds new image → Pushes to ACR → Updates manifest.
   * Argo CD detects changes → Deploys to AKS.
2. **Troubleshooting**:
   * **Image Pull Errors**: Ensure ACR credentials are added to AKS:

kubectl create secret docker-registry acr-secret \

--docker-server=azure-cicd-acr.azurecr.io \

--docker-username=<ACR-USERNAME> \

--docker-password=<ACR-PASSWORD>

* + **Argo CD Sync Issues**: Check logs with kubectl logs -n argocd -l app.kubernetes.io/name=argocd-server.