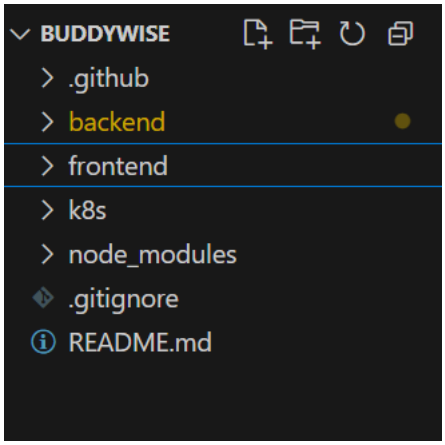
Buddy Wise Application Deployment

This project automates the **build**, **deployment**, and **infrastructure provisioning** of a cloud-native application using:

- **GitHub Actions** for CI/CD
- **Docker** for containerization
- **Kubernetes (AKS)** for orchestration
- **Azure** for cloud infrastructure
- **Terraform** for Infrastructure-as-Code (IaC)

Project Structure



Docker Setup

Frontend Dockerfile

Build-Deploy.yml | Dockerfile 1

frontend > todo > Dockerfile > ...

```
# Use an official Node.js image as base
FROM node:18-alpine

# Set the working directory inside the container
WORKDIR /frontend

# Copy only package.json and package-lock.json first
COPY package.json package-lock.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application code
COPY . .

# Build the React app (optional, if needed)
# RUN npm run build

# Expose the port your app runs on (e.g., 3000 for React)
EXPOSE 3000

# Start the app
CMD ["npm", "start"]

```

Backend Dockerfile

Build-Deploy.yml | Dockerfile 1

backend > Dockerfile > ...

```dockerfile
# Use Python as base image
FROM python:3.11

# Set working directory inside the container
WORKDIR /backend

# Copy dependency file first
COPY requirements.txt ./

# Install dependencies
RUN pip install --no-cache-dir --upgrade pip && \
    pip install --no-cache-dir -r requirements.txt

# Copy the rest of the application files
COPY . .

# Expose the FastAPI default port
EXPOSE 8501

# Start the application with Uvicorn
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8501"]
```

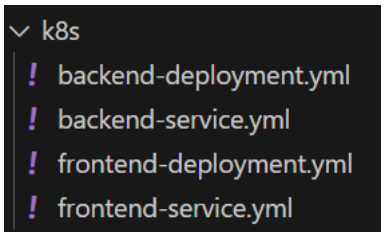
Kubernetes Setup

Frontend

- k8s/frontend-deployment.yml
- k8s/frontend-service.yml

Backend

- k8s/backend-deployment.yml
- k8s/backend-service.yml



If you want to apply directly from the CLI:

kubectl apply -f k8s/

After deployment Completion, need to run the following commands to know the Pods' Status

$> buddywise-infra> kubectl get deployments -n prod

```
PS D:\Buddy\buddywise-infra\buddywise-infra> kubectl get deployments -n prod
NAME              READY   UP-TO-DATE   AVAILABLE   AGE
myapp-backend     2/2     2            2           2d6h
myapp-frontend    2/2     2            2           2d6h
```

$> buddywise-infra> kubectl describe pod myapp-frontend -n prod

```
PS D:\Buddy\buddywise-infra\buddywise-infra> kubectl describe pod myapp-frontend -n prod
Name:             myapp-frontend-6bdf5449db-82fvs
Namespace:        prod
Priority:         0
Service Account:  default
Node:             aks-agentpool-40028103-vmss000000/10.224.0.4
Start Time:       Wed, 09 Apr 2025 08:06:06 +0200
Labels:           app=myapp-frontend
                  pod-template-hash=6bdf5449db
Annotations:      <none>
Status:           Running
IP:               10.244.1.110
IPs:
  IP:           10.244.1.110
Controlled By:  ReplicaSet/myapp-frontend-6bdf5449db
Containers:
  myapp-frontend:
    Container ID:   containerd://0ad3990e26f3178b2cd0c456a85f20541ad59e9a7f3e9ce803c8fdc8a359a8a8
    Image:          acrbuddywise.azurecr.io/myapp-frontend:latest
    Image ID:       acrbuddywise.azurecr.io/myapp-frontend@sha256:05b4d5e1cfe5b35ccc34e1e647747364d33147d65993ff76ea
cee567f74c6625
    Port:           3000/TCP
    Host Port:      0/TCP
```

```
    State:          Running
      Started:      Wed, 09 Apr 2025 08:06:17 +0200
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-bnh7s (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-bnh7s:
    Type:                     Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:   3607
    ConfigMapName:            kube-root-ca.crt
    ConfigMapOptional:        <nil>
    DownwardAPI:              true
QoS Class:                    BestEffort
Node-Selectors:               <none>
Tolerations:                  node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:                       <none>

Name:             myapp-frontend-6bdf5449db-8h2bd
Namespace:        prod
Priority:         0
Service Account:  default
Node:             aks-agentpool-40028103-vmss000001/10.224.0.5
Start Time:       Wed, 09 Apr 2025 08:06:06 +0200
```

```
Labels:           app=myapp-frontend
                  pod-template-hash=6bdf5449db
Annotations:      <none>
Status:           Running
IP:               10.244.0.56
IPs:
  IP:           10.244.0.56
Controlled By:  ReplicaSet/myapp-frontend-6bdf5449db
Containers:
  myapp-frontend:
    Container ID:   containerd://ccaeffb888f8ff8dfbec00d3b0e18eaf1c201048b524f5bb2a22b1db522261ca
    Image:          acrbuddywise.azurecr.io/myapp-frontend:latest
    Image ID:       acrbuddywise.azurecr.io/myapp-frontend@sha256:05b4d5e1cfe5b35ccc34e1e647747364d33147d65993ff76e
cee567f74c6625
    Port:           3000/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Wed, 09 Apr 2025 08:06:17 +0200
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-c2bx9 (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-c2bx9:
    Type:                     Projected (a volume that contains injected data from multiple sources)
```

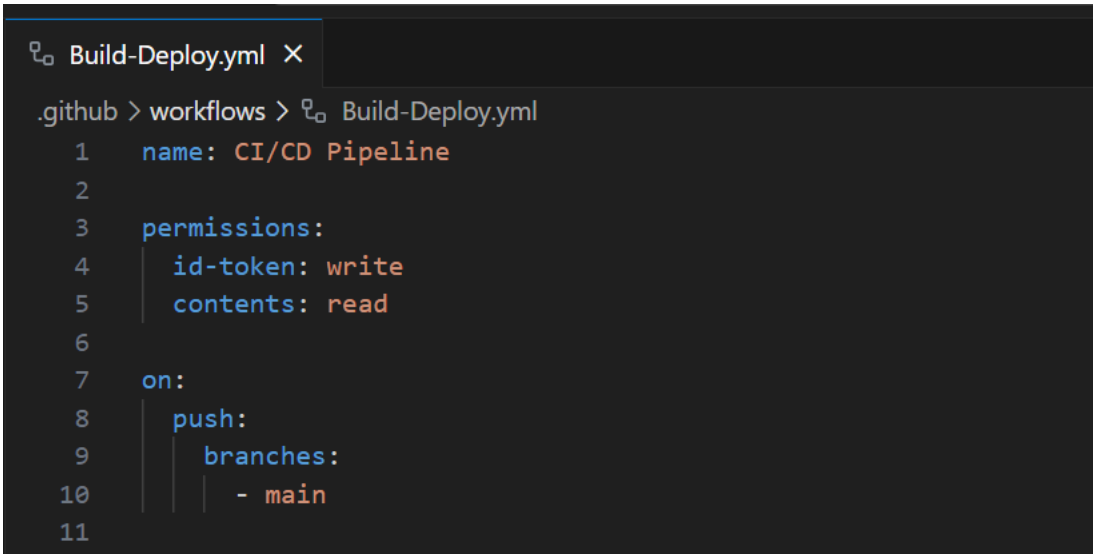
```
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:                      <none>

PS D:\Buddy\buddywise-infra\buddywise-infra>
```

GitHub Actions Workflow

.github/workflows/deployment.yml

This workflow automates:

- Docker image build
    - Tiger
    - 

- Azure login using OIDC & Client Secret & Job

- Build-Deploy.yml

.github > workflows > Build-Deploy.yml

```yaml

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Log in to Azure
        uses: azure/login@v1
        with:
          client-id: ${{ secrets.AZURE_CLIENT_ID }}
          tenant-id: ${{ secrets.AZURE_TENANT_ID }}
          subscription-id: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
          creds: ${{ secrets.AZURE_CREDENTIALS }}
          #enable-AzPSSession: true
          #enable-AzPSSession: false
          auth-type: SERVICE_PRINCIPAL
          #allow-no-subscription: true

      - name: Log in to Azure Container Registry
        uses: azure/docker-login@v1
        with:
          login-server: ${{ secrets.REGISTRY_LOGIN_SERVER }}
          username: ${{ secrets.AZURE_CLIENT_ID }}
          password: ${{ secrets.AZURE_CLIENT_SECRET }}

      - name: Build Docker images
```

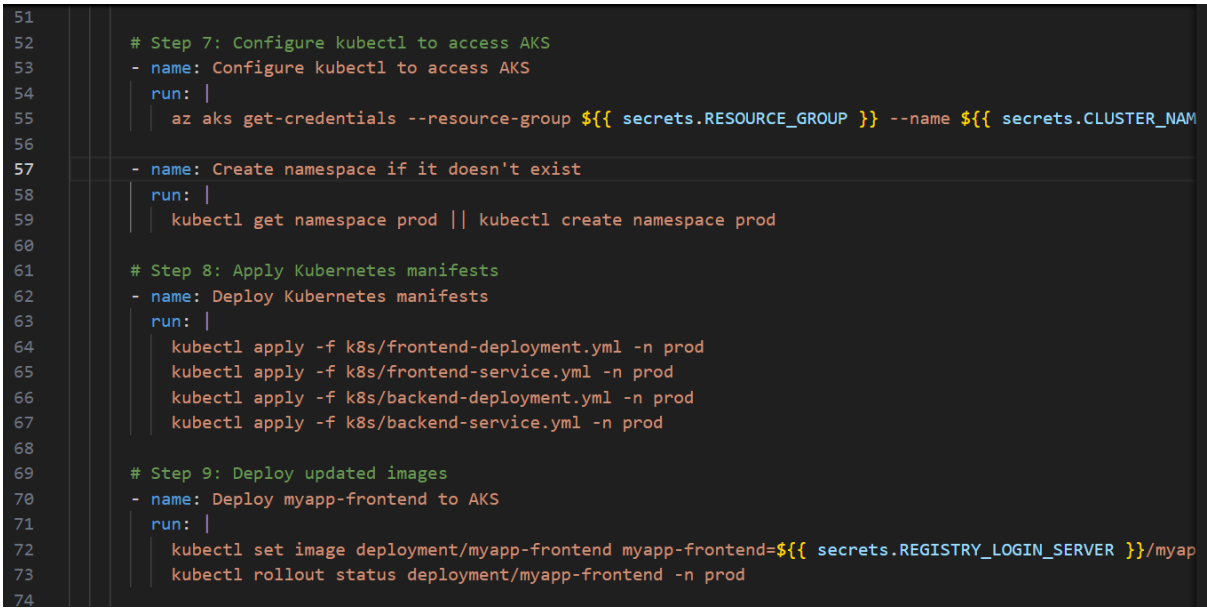
- Pushing to ACR

```yaml
    - name: Build Docker images
      run: |
        docker build -t myapp-backend:latest ./backend
        docker build -f frontend/todo/Dockerfile -t myapp-frontend:latest ./frontend/todo
        docker tag myapp-backend:latest ${{ secrets.REGISTRY_LOGIN_SERVER }}/myapp-backend:latest
        docker tag myapp-frontend:latest ${{ secrets.REGISTRY_LOGIN_SERVER }}/myapp-frontend:latest

    - name: Push Docker images to ACR
      run: |
        az acr login --name acrbuddywise.azurecr.io
        docker push ${{ secrets.REGISTRY_LOGIN_SERVER }}/myapp-backend:latest
        docker push ${{ secrets.REGISTRY_LOGIN_SERVER }}/myapp-frontend:latest

```

- Deployment to AKS



```
51
52      # Step 7: Configure kubectl to access AKS
53      - name: Configure kubectl to access AKS
54        run: |
55          az aks get-credentials --resource-group ${{ secrets.RESOURCE_GROUP }} --name ${{ secrets.CLUSTER_NAM
56
57      - name: Create namespace if it doesn't exist
58        run: |
59          kubectl get namespace prod || kubectl create namespace prod
60
61      # Step 8: Apply Kubernetes manifests
62      - name: Deploy Kubernetes manifests
63        run: |
64          kubectl apply -f k8s/frontend-deployment.yml -n prod
65          kubectl apply -f k8s/frontend-service.yml -n prod
66          kubectl apply -f k8s/backend-deployment.yml -n prod
67          kubectl apply -f k8s/backend-service.yml -n prod
68
69      # Step 9: Deploy updated images
70      - name: Deploy myapp-frontend to AKS
71        run: |
72          kubectl set image deployment/myapp-frontend myapp-frontend=${{ secrets.REGISTRY_LOGIN_SERVER }}/myap
73          kubectl rollout status deployment/myapp-frontend -n prod
74
```

```
- name: Deploy myapp-backend to AKS
  run: |
    kubectl set image deployment/myapp-backend myapp-backend=${{ secrets.REGISTRY_LOGIN_SERVER }}/myapp-backen
    kubectl rollout status deployment/myapp-backend -n prod
```

GitHub Secrets Configuration:

Add the following in GitHub → Settings → Secrets and Variables:

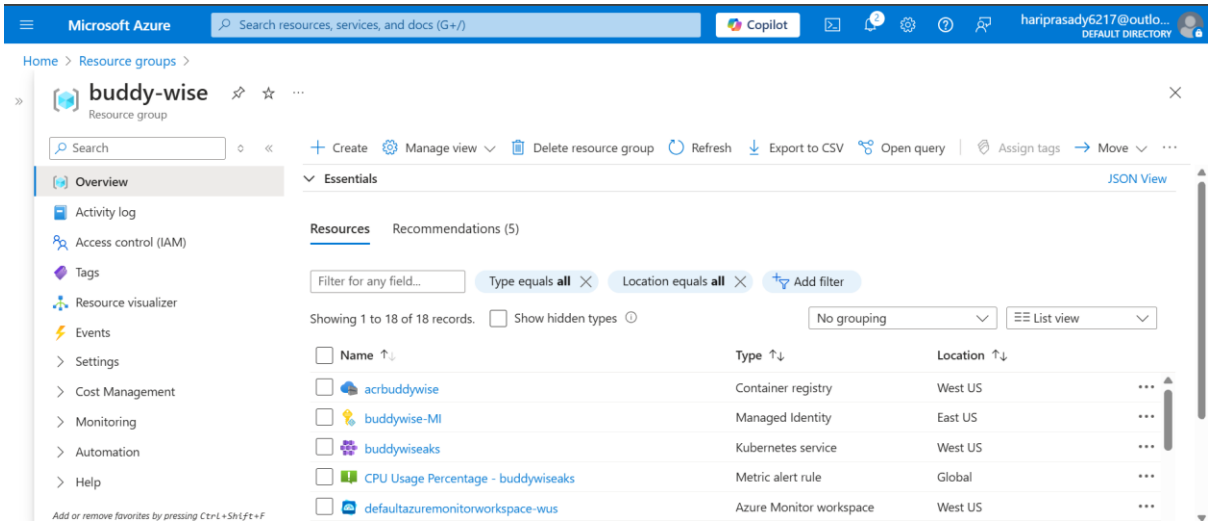
Secret Name	Purpose
AZURE_CLIENT_ID	App Registration Client ID
AZURE_CLIENT_SECRET	App Registration Secret
AZURE_TENANT_ID	Azure Tenant ID
AZURE_SUBSCRIPTION_ID	Subscription ID
AZURE_RESOURCE_GROUP	Resource Group Name
AZURE_ACR_NAME	ACR Name

Azure Infrastructure Provisioning (UI-Based):

Create Resource Group

Azure Portal → "Resource groups" → **Create**

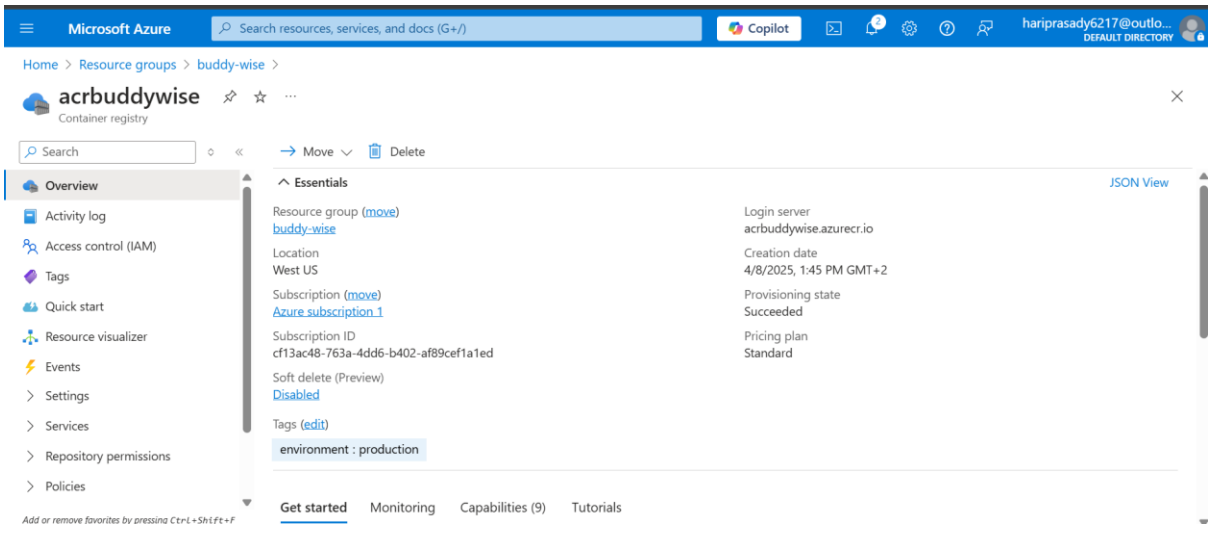
- Resource group: buddywise-rg
- Region: East US



Create Azure Container Registry (ACR)

Azure Portal → "Container registries" → **Create**

- Name: buddywiseacr (must be unique)
- SKU: Standard
- Enable Admin User: yes

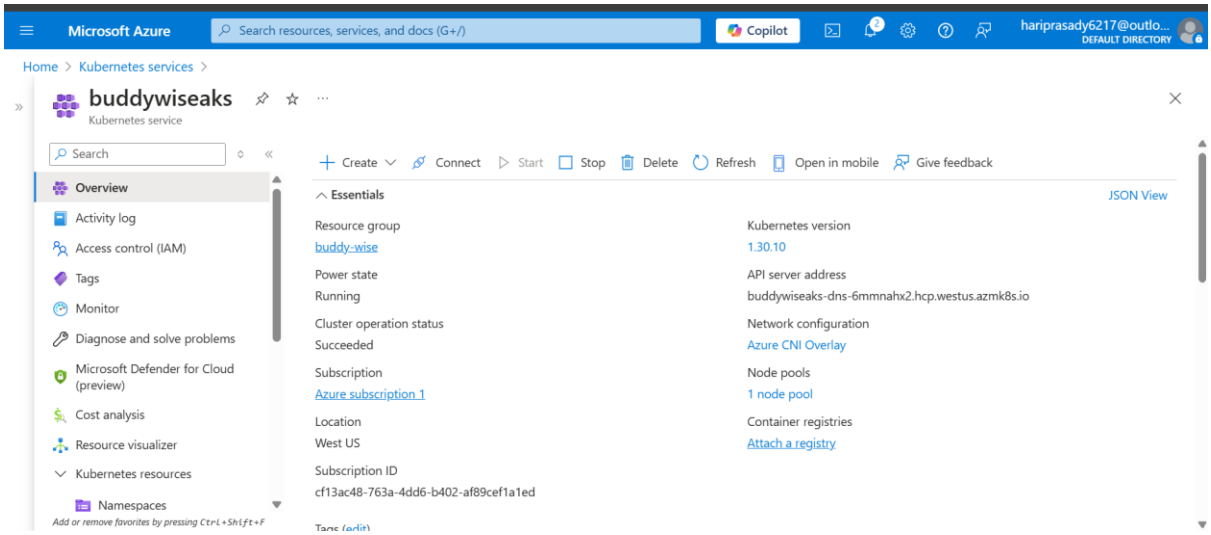


Create Azure Kubernetes Cluster (AKS):

Azure Portal → "Kubernetes services" → **Create**

- Name: buddywiseaks

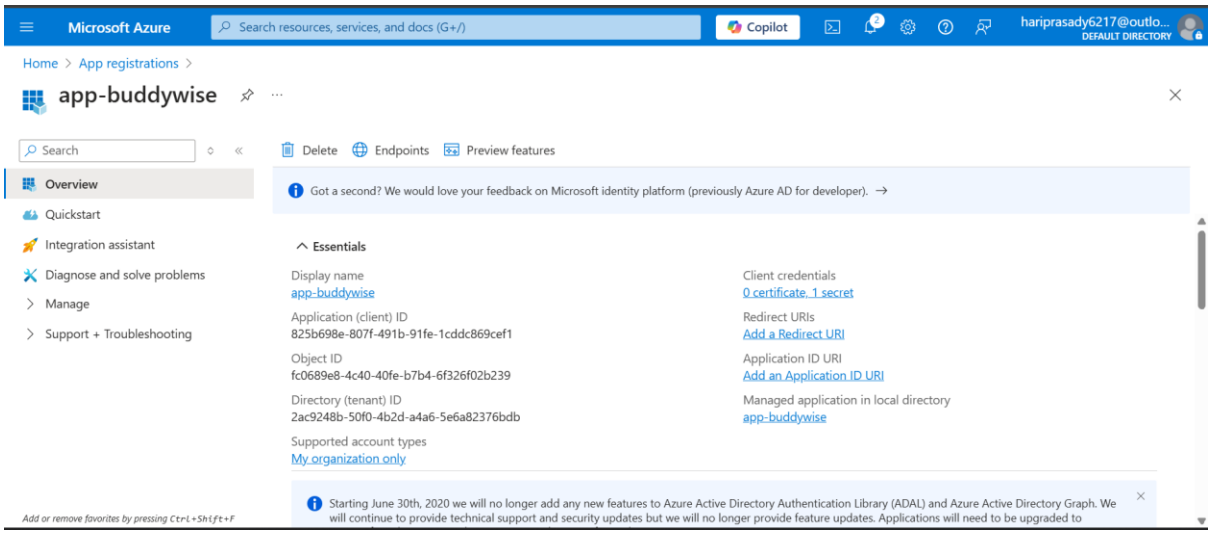
- Node Pool: 2 nodes, Standard_DS2_v2
- Auth: System-assigned managed identity
- Integrate with ACR: buddywiseacr



Create App Registration:

Azure Portal → "Microsoft Entra ID" → "App registrations" → **New registration**

- Name: buddywise-gh-actions
- Save Application (client) ID

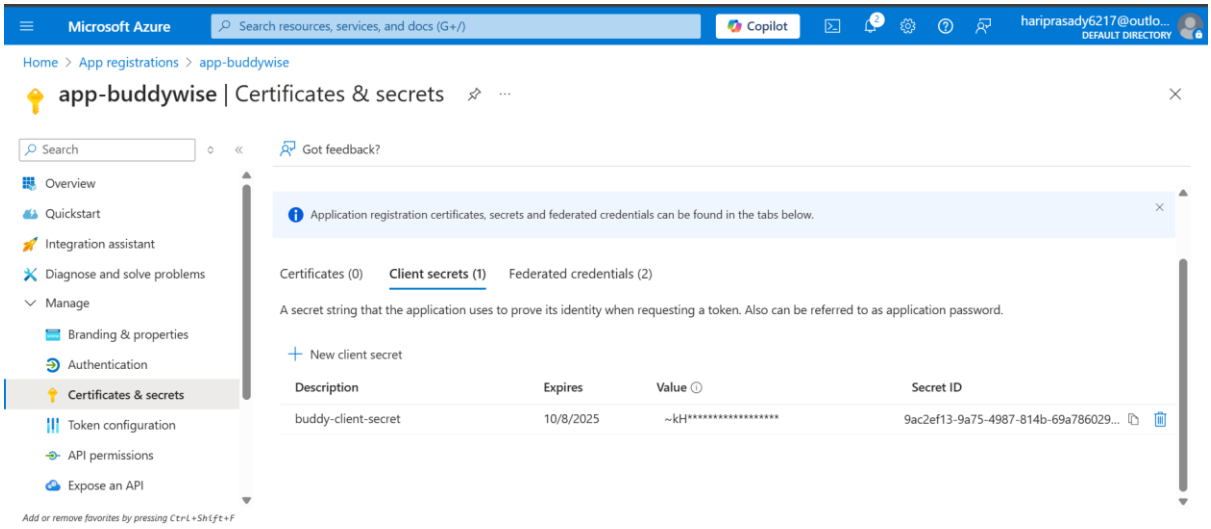


Create Client Secret:

App → "Certificates & secrets" → New client secret

Save the value (used in GitHub Secrets)

And save value, it appears only once



Assign IAM Roles:

- **AKS Access**

AKS → Access control (IAM) → Add role:

- Role: Contributor
- Assign to: Service Principal

- **ACR Access**

ACR → Access control (IAM):

- Role: AcrPush or AcrPull

Configure OIDC Federated Identity (Optional, Preferred):

az ad app federated-credential create --id "825b698e-807f-491b-91fe-1cddc869cef1" --parameters "{`"name`": `"github-actions`", `"issuer`": `"https://token.actions.githubusercontent.com`", `"subject`": `"repo:hariprasadyadagiri/Buddywise:ref:refs/heads/main`", `"description`": `"OIDC for GitHub Actions`", `"audiences`": [`"api://AzureADTokenExchange`"]}"

All files and access have been completed. Let's deploy the application from Git.

Use the commands below from the CLI to deploy from git

:> git add.

:> git commit -m "Deployment Application" (you use your own massage while committing)

:> git push origin main