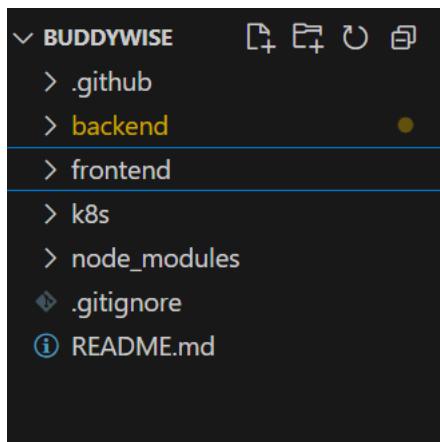


Buddy Wise Application Deployment

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

The screenshot shows a code editor interface with two tabs: "Build-Deploy.yml" and "Dockerfile 1". The "Dockerfile 1" tab is active and displays the following Dockerfile content:

```
1 # Use an official Node.js image as base
2 FROM node:18-alpine
3
4 # Set the working directory inside the container
5 WORKDIR /frontend
6
7 # Copy only package.json and package-lock.json first
8 COPY package.json package-lock.json ./
9
10 # Install dependencies
11 RUN npm install
12
13 # Copy the rest of the application code
14 COPY . .
15
16 # Build the React app (optional, if needed)
17 # RUN npm run build
18
19 # Expose the port your app runs on (e.g., 3000 for React)
20 EXPOSE 3000
21
22 # Start the app
23 CMD ["npm", "start"]
24
```

Backend Dockerfile

The screenshot shows a code editor interface with two tabs. The active tab is 'Dockerfile 1' containing a Dockerfile for a Python application. The Dockerfile includes commands for setting the base image, working directory, copying dependencies, installing pip, exposing port 8501, and running Uvicorn. The second tab, 'Build-Deploy.yml', is visible but its content is not shown.

```
backend > Dockerfile > ...
1 # Use Python as base image
2 FROM python:3.11
3
4 # Set working directory inside the container
5 WORKDIR /backend
6
7 # Copy dependency file first
8 COPY requirements.txt .
9
10 # Install dependencies
11 RUN pip install --no-cache-dir --upgrade pip && \
12     pip install --no-cache-dir -r requirements.txt
13
14 # Copy the rest of the application files
15 COPY .
16
17 # Expose the FastAPI default port
18 EXPOSE 8501
19
20 # Start the application with Uvicorn
21 CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8501"]
22
```

Kubernetes Setup

Frontend

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

Backend

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

The screenshot shows a file browser interface with a 'k8s' folder expanded. Inside the 'k8s' folder are four files: backend-deployment.yml, backend-service.yml, frontend-deployment.yml, and frontend-service.yml, each preceded by a red exclamation mark indicating they are excluded from the current view.

```
✓ k8s
! backend-deployment.yml
! backend-service.yml
! frontend-deployment.yml
! frontend-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@sha256:05b4d5e1cf5b35ccc34e1e647747364d33147d65993ff76ea
      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:
  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@sha256:05b4d5e1cfef5b35ccc34e1e647747364d33147d65993ff76e
    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>

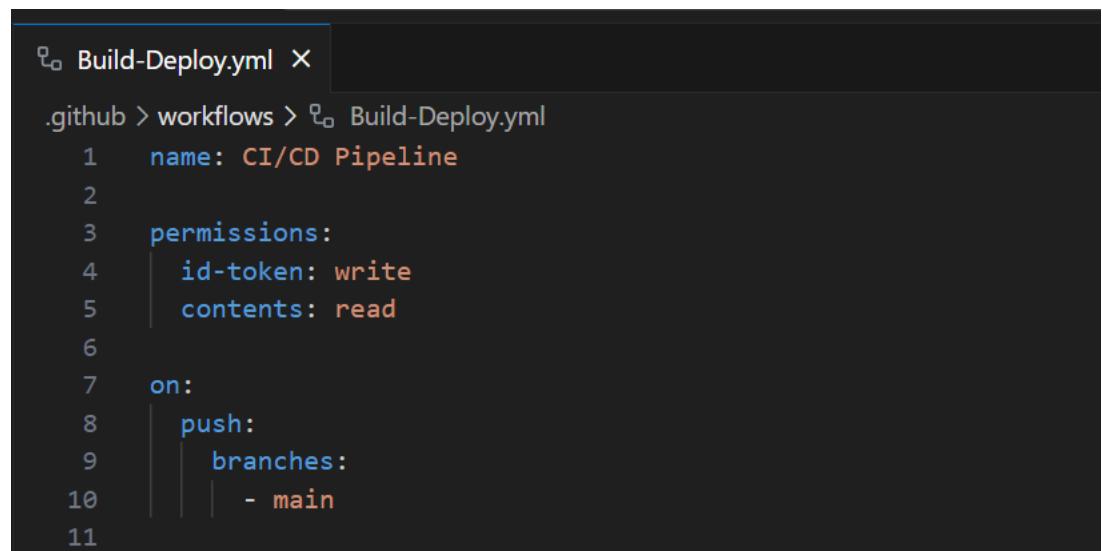
○ PS D:\Buddy\buddywise-infra\buddywise-infra> []
```

GitHub Actions Workflow

.github/workflows/deployment.yml

This workflow automates:

- Docker image build
 - Tiger



The screenshot shows a GitHub code editor with the file ".github/workflows/Build-Deploy.yml" open. The code defines a CI/CD pipeline with permissions and triggers for pushes to the main branch.

```
name: CI/CD Pipeline
permissions:
  id-token: write
  contents: read
on:
  push:
    branches:
      - main
```

- Azure login using OIDC & Client Secret & Job

```
Build-Deploy.yml X
.github > workflows > Build-Deploy.yml
11
12   jobs:
13     build:
14       runs-on: ubuntu-latest
15
16     steps:
17       - name: Checkout code
18         uses: actions/checkout@v3
19
20       - name: Log in to Azure
21         uses: azure/login@v1
22         with:
23           client-id: ${{ secrets.AZURE_CLIENT_ID }}
24           tenant-id: ${{ secrets.AZURE_TENANT_ID }}
25           subscription-id: ${{ secrets.AZURE_SUBSCRIPTION_ID }}
26           creds: ${{ secrets.AZURE_CREDENTIALS }}
27           #enable-AzPSSession: true
28           #enable-AzPSSession: false
29           auth-type: SERVICE_PRINCIPAL
30           #allow-no-subscription: true
31
32       - name: Log in to Azure Container Registry
33         uses: azure/docker-login@v1
34         with:
35           login-server: ${{ secrets.REGISTRY_LOGIN_SERVER }}
36           username: ${{ secrets.AZURE_CLIENT_ID }}
37           password: ${{ secrets.AZURE_CLIENT_SECRET }}
38
39       - name: Build Docker images

```

- Pushing to ACR

```
9   - name: Build Docker images
10    run: |
11      docker build -t myapp-backend:latest ./backend
12      docker build -f frontend/todo/Dockerfile -t myapp-frontend:latest ./frontend/todo
13      docker tag myapp-backend:latest ${{ secrets.REGISTRY_LOGIN_SERVER }}}/myapp-backend:latest
14      docker tag myapp-frontend:latest ${{ secrets.REGISTRY_LOGIN_SERVER }}}/myapp-frontend:latest
15
16   - name: Push Docker images to ACR
17    run: |
18      az acr login --name acrbuddywise.azurecr.io
19      docker push ${{ secrets.REGISTRY_LOGIN_SERVER }}}/myapp-backend:latest
20      docker push ${{ secrets.REGISTRY_LOGIN_SERVER }}}/myapp-frontend:latest
21
```

- Deployment to AKS

```

51      # Step 7: Configure kubectl to access AKS
52      - name: Configure kubectl to access AKS
53          run: |
54              az aks get-credentials --resource-group ${{ secrets.RESOURCE_GROUP }} --name ${{ secrets.CLUSTER_NAME }}
55
56
57      - name: Create namespace if it doesn't exist
58          run: |
59              kubectl get namespace prod || kubectl create namespace prod
60
61      # Step 8: Apply Kubernetes manifests
62      - name: Deploy Kubernetes manifests
63          run: |
64              kubectl apply -f k8s/frontend-deployment.yml -n prod
65              kubectl apply -f k8s/frontend-service.yml -n prod
66              kubectl apply -f k8s/backend-deployment.yml -n prod
67              kubectl apply -f k8s/backend-service.yml -n prod
68
69      # Step 9: Deploy updated images
70      - name: Deploy myapp-frontend to AKS
71          run: |
72              kubectl set image deployment/myapp-frontend myapp-frontend=${{ secrets.REGISTRY_LOGIN_SERVER }}/myap
73                  kubectl rollout status deployment/myapp-frontend -n prod
74
75
76      - name: Deploy myapp-backend to AKS
77          run: |
78              kubectl set image deployment/myapp-backend myapp-backend=${{ secrets.REGISTRY_LOGIN_SERVER }}/myapp-backen
79                  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: buddywseacr (must be unique)
- SKU: Standard
- Enable Admin User: yes

Create Azure Kubernetes Cluster (AKS):

Azure Portal → "Kubernetes services" → Create

- Name: buddywseaks

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

buddywiseaks Kubernetes service

Overview

Essentials

- Resource group: buddy-wise
- Kubernetes version: 1.30.10
- Power state: Running
- Cluster operation status: Succeeded
- Subscription: Azure subscription 1
- Location: West US
- Subscription ID: cf13ac48-763a-4dd6-b402-af89cef1a1ed
- Network configuration: Azure CNI Overlay
- Node pools: 1 node pool
- Container registries: Attach a registry

Create App Registration:

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

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

app-buddywise

Overview

Essentials

- Display name: app-buddywise
- Application (client) ID: 825b698e-807f-491b-91fe-1cddc869cef1
- Object ID: fc0689e8-4c40-40fe-b7b4-6f326f02b239
- Directory (tenant) ID: 2ac9248b-50f0-4b2d-a4a6-5e6a82376bdb
- Supported account types: My organization only
- Client credentials: 0_certificate_1_secret
- Redirect URLs: Add a Redirect URI
- Application ID URI: Add an Application ID URI
- Managed application in local directory: app-buddywise

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure Active Directory Graph. We will continue to provide technical support and security updates but we will no longer provide feature updates. Applications will need to be upgraded to ...

Create Client Secret:

App → "Certificates & secrets" → New client secret

Save the value (used in GitHub Secrets)

And save value, it appears only once

The screenshot shows the Microsoft Azure portal interface for managing app registrations. The left sidebar has a 'Certificates & secrets' section selected. In the main content area, there is a table listing a single client secret named 'buddy-client-secret'. The table columns are Description, Expires, Value (redacted), and Secret ID.

Description	Expires	Value	Secret ID
buddy-client-secret	10/8/2025	~kH*****	9ac2ef13-9a75-4987-814b-69a786029...

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 message while committing)

:> git push origin main

