Project Plan: Containerization and Deployment of Wisecow Application on Kubernetes

1. \*Dockerization\*

\*Dockerfile Creation\*

- Create a Dockerfile to containerize the Wisecow application.

- Ensure the Dockerfile includes the necessary base image, dependencies, and commands to run the application.

\*Example Dockerfile:\*

dockerfile

\* Use a suitable base image

FROM node:14

# Set the working directory

WORKDIR /usr/src/app

# Copy package.json and install dependencies

COPY package\*.json ./

RUN npm install

# Copy the application source code

COPY . .

# Expose the application port

EXPOSE 3000

# Command to run the application

CMD ["npm", "start"]

### 2. \*Kubernetes Deployment\*

#### \*Kubernetes Manifest Files\*

- Create deployment and service manifest files to deploy the application in a Kubernetes environment.

\*\*Example Deployment Manifest (wisecow-deployment.yaml):\*\*

yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: wisecow

spec:

replicas: 3

selector:

matchLabels:

app: wisecow

template:

metadata:

labels:

app: wisecow

spec:

containers:

- name: wisecow

image: <your-docker-registry>/wisecow:latest

ports:

- containerPort: 3000

\*\*Example Service Manifest (wisecow-service.yaml):\*\*

yaml

apiVersion: v1

kind: Service

metadata:

name: wisecow

spec:

type: ClusterIP

ports:

- port: 80

targetPort: 3000

selector:

app: wisecow

### 3. \*Continuous Integration and Deployment (CI/CD)\*

#### \*GitHub Actions Workflow\*

- Implement a GitHub Actions workflow to automate the build and push of the Docker image and deploy the application.

\*\*Example GitHub Actions Workflow (.github/workflows/ci-cd.yml):\*\*

yaml

name: CI/CD Pipeline

on:

push:

branches:

- main

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Build Docker image

run: |

docker build -t <your-docker-registry>/wisecow:latest .

- name: Login to Docker Registry

run: echo "${{ secrets.DOCKER\_PASSWORD }}" | docker login -u "${{ secrets.DOCKER\_USERNAME }}" --password-stdin

- name: Push Docker image

run: docker push <your-docker-registry>/wisecow:latest

deploy:

runs-on: ubuntu-latest

needs: build

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Setup kubectl

uses: azure/setup-kubectl@v1

with:

version: 'latest'

- name: Deploy to Kubernetes

run: |

kubectl apply -f wisecow-deployment.yaml

kubectl apply -f wisecow-service.yaml

### 4. \*TLS Implementation\*

- Configure TLS for the application to ensure secure communication.

- You may use tools like Certbot or Kubernetes Secrets to manage TLS certificates.

\*Example Kubernetes Secret for TLS:\*

yaml

apiVersion: v1

kind: Secret

metadata:

name: wisecow-tls

type: kubernetes.io/tls

data:

tls.crt: <base64-encoded-cert>

tls.key: <base64-encoded-key>

### 5. \*Repository Setup\*

- Create a private GitHub repository containing:

- Wisecow application source code.

- Dockerfile.

- Kubernetes manifest files.

- GitHub Actions workflow file.

### 6. \*Access Control\*

- Set the GitHub repository to public once completed, ensuring that all configurations are properly tested and documented.

### \*End Goal\*

Achieve successful containerization and deployment of the Wisecow application on Kubernetes with automated CI/CD and secure TLS communication.

### \*Next Steps\*

- Implement and test each step iteratively.

- Validate TLS communication and overall application functionality post-deployment.