Book-My-Show Application - Script File code

1. Jenkinsfile Script:

→ To implement the CI/CD pipeline using a Jenkinsfile.

```
pipeline {
    agent any
    tools {
        jdk 'jdk21'
        maven 'maven3'
    }
    environment {
        SCANNER_HOME = tool 'sonar'
        DOCKER_IMAGE = 'bookmyshow-app'
        DOCKER_HUB_REPO = 'arjukumar7/bookmyshow-app'
        IMAGE_TAG = "v${BUILD_NUMBER}"
    }

    stages {
        stage('Clean Workspace') {
            steps {
                cleanWs()
            }
        }
        stage('Checkout from Git') {
            steps {
                git branch: 'main', url: https://github.com/arju7jha/Book-My-Show.git
                sh 'ls -la'
            }
        }
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('sonar') {
                    sh '''
                        $SCANNER_HOME/bin/sonar-scanner \
                            -Dsonar.projectName=arju-BookMyShow-app \
```

```
                    -Dsonar.projectKey=arju
                '''
            }
        }
    }
    stage('Quality Gate') {
        steps {
            script {
                timeout(time: 2, unit: 'MINUTES') {
                    try {
                        waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
                    } catch (Exception e) {
                        echo "Quality Gate timeout - Check SonarQube dashboard manually"
                        echo "SonarQube Dashboard: http://localhost:9001/dashboard?id=arju"
                    }
                }
            }
        }
    }
    stage('Build Docker Image') {
        steps {
            dir('bookmyshow-app') {
                sh '''
                    docker build -t ${DOCKER_IMAGE}:${IMAGE_TAG} .
                    docker tag ${DOCKER_IMAGE}:${IMAGE_TAG}
${DOCKER_HUB_REPO}:${IMAGE_TAG}
                '''
            }
        }
    }
    stage('Push to Docker Hub') {
        steps {
            script {
                withCredentials([string(credentialsId: 'docker-hub-token', variable: 'DOCKER_TOKEN')]) {
                    sh '''
                        echo $DOCKER_TOKEN | docker login -u arjukumar7 --password-stdin
                        docker push ${DOCKER_HUB_REPO}:${IMAGE_TAG}
```

```
                    docker logout
                '''
            }
        }
    }
    stage('Deploy Container') {
        steps {
            sh '''
                docker stop bookmyshow-app-run || true
                docker rm bookmyshow-app-run || true
                docker run -d -p 3000:80 --name bookmyshow-app-run
${DOCKER_HUB_REPO}:${IMAGE_TAG}
            '''
        }
    }
  }
  post {
    always {
        echo "Pipeline completed. SonarQube analysis results available at:
http://localhost:9001/dashboard?id=arju"
        echo "Docker image pushed: ${DOCKER_HUB_REPO}:${IMAGE_TAG}"
        echo "Application running at: http://localhost:3000"
    }
    success {
        sh '''
            docker rmi ${DOCKER_IMAGE}:${IMAGE_TAG} || true
        '''
        emailext (
            subject: "SUCCESS: BookMyShow CI/CD Pipeline Completed",
            body: """
            Hi Team,

            The BookMyShow CI/CD pipeline completed successfully.

            Pipeline Details:
            - SonarQube Analysis: Completed
```

```
            - Docker Image: ${DOCKER_HUB_REPO}:${IMAGE_TAG}
            - Application URL: http://localhost:3000

            You can view the SonarQube report here:
            http://172.17.0.2:9000/dashboard?id=arju

            Regards,
            Jenkins
            """,
            to: 'arjuk217@gmail.com'
        )
    }
    failure {
        emailext (
            subject: "FAILED: BookMyShow CI/CD Pipeline",
            body: """
            Hi Team,

            The BookMyShow CI/CD pipeline has failed.

            Please check the Jenkins console output for details.
            SonarQube Dashboard: http://172.17.0.2:9000/dashboard?id=arju

            Regards,
            Jenkins
            """,
            to: 'arjuk217@gmail.com'
        )
    }
  }
}
```

2. Jenkinsfile Script for Docker Push :

→ Docker image push to DockerHub through the Jenkins pipeline (not manually).

```
pipeline {
    agent any
    environment {
        IMAGE_NAME = "arjukumar7/bookmyshow-app:v1"
    }
    stages {
        stage('Checkout') {
            steps {
                git branch: 'feature', url: 'https://github.com/arju7jha/Book-My-Show.git'
            }
        }
        stage('Build Docker Image') {
            steps {
                bat 'docker build --no-cache -t %IMAGE_NAME% ./bookmyshow-app'
            }
        }
        stage('Push to DockerHub') {
            steps {
                withCredentials([usernamePassword(credentialsId: 'dockerhub-cred',
usernameVariable: 'DOCKER_USER', passwordVariable: 'DOCKER_PASS')]) {
                    bat """
                    docker login -u %DOCKER_USER% -p %DOCKER_PASS%
                    docker push %IMAGE_NAME%
                    """
                }
            }
        }
    }
}
```

3. Dockerfile Script :

→ Dockerfile to build and expose port for the application .

```
# Use Node.js 18 (or your Jenkins-configured version)
FROM node:18

# Set working directory ----------------
WORKDIR /app

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

# Install chokidar@3 explicitly (fix for missing dependency)
RUN npm install chokidar@3 --legacy-peer-deps

# Your existing postcss fix
RUN npm install postcss@8.4.21 postcss-safe-parser@6.0.0 --legacy-peer-deps

# Install rest of the dependencies
RUN npm install --legacy-peer-deps

# Copy the entire project
COPY . .

# Expose port 3000
EXPOSE 3000

# Set environment variable to prevent OpenSSL errors
ENV NODE_OPTIONS=--openssl-legacy-provider
ENV PORT=3000

# Start the application
CMD ["npm", "start"]
```

4. Kubernetes Manifests Script files :

→ Kubernetes YAML files (`deployment.yaml`, `service.yaml`).

deployment.yaml :

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bookmyshow-deployment
  namespace: arju
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bookmyshow
  template:
    metadata:
      labels:
        app: bookmyshow
    spec:
      containers:
      - name: bookmyshow
        image: arjukumar7/bookmyshow-app:v1
        ports:
        - containerPort: 3000
```

service.yaml :

```
apiVersion: v1
kind: Service
metadata:
  name: bookmyshow-service
  namespace: arju
spec:
  type: LoadBalancer
  selector:
    app: bookmyshow
  ports:
    - protocol: TCP
      port: 80
      targetPort: 3000
```