DevOps Shack

Production Grade CICD Pipeline

With Explanation

```
pipeline {
    agent any

    tools {
        jdk 'jdk17'
        maven 'maven3'
    }

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
    }

    stages {
        stage('Git Checkout') {
            steps {
                git branch: 'main', credentialsId: 'git-cred', url:
'https://github.com/jaiswaladi246/Boardgame.git'
            }
        }

        stage('Compile') {
            steps {
                sh "mvn compile"
            }
```

```
    }

    stage('Test') {
        steps {
            sh "mvn test"
        }
    }

    stage('File System Scan') {
        steps {
            sh "trivy fs --format table -o trivy-fs-report.html ."
        }
    }

    stage('SonarQube Analysis') {
        steps {
            withSonarQubeEnv('sonar') {
                sh """$SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=BoardGame -Dsonar.projectKey=BoardGame \
                    -Dsonar.java.binaries='.'"""
            }
        }
    }

    stage('Quality Gate') {
        steps {
            script {
                waitForQualityGate abortPipeline: false, credentialsId: 'sonar-token'
            }
        }
    }

    stage('Build') {
        steps {
            sh "mvn package"
        }
    }

    stage('Publish To Nexus') {
        steps {
```

```
            withMaven(globalMavenSettingsConfig: 'global-settings', jdk: 'jdk17',
maven: 'maven3', mavenSettingsConfig: '', traceability: true) {
                sh "mvn deploy"
            }
        }
    }

    stage('Build & Tag Docker Image') {
        steps {
            script {
                withDockerRegistry(credentialsId: 'docker-cred', toolName:
'docker') {
                    sh "docker build -t adijaiswal/boardshack:latest ."
                }
            }
        }
    }

    stage('Docker Image Scan') {
        steps {
            sh "trivy image --format table -o trivy-image-report.html
adijaiswal/boardshack:latest"
        }
    }

    stage('Push Docker Image') {
        steps {
            script {
                withDockerRegistry(credentialsId: 'docker-cred', toolName:
'docker') {
                    sh "docker push adijaiswal/boardshack:latest"
                }
            }
        }
    }

    stage('Deploy To Kubernetes') {
        steps {
```

```
            withKubeConfig(caCertificate: '', clusterName: 'kubernetes',
contextName: '', credentialsId: 'k8-cred', namespace: 'webapps',
restrictKubeConfigAccess: false, serverUrl: 'https://172.31.8.146:6443') {
                sh "kubectl apply -f deployment-service.yaml"
            }
        }
      }

      stage('Verify the Deployment') {
         steps {
            withKubeConfig(caCertificate: '', clusterName: 'kubernetes',
contextName: '', credentialsId: 'k8-cred', namespace: 'webapps',
restrictKubeConfigAccess: false, serverUrl: 'https://172.31.8.146:6443') {
                sh "kubectl get pods -n webapps"
                sh "kubectl get svc -n webapps"
            }
         }
      }
   }

   post {
      always {
         script {
            def jobName = env.JOB_NAME
            def buildNumber = env.BUILD_NUMBER
            def pipelineStatus = currentBuild.result ?: 'UNKNOWN'
            def bannerColor = pipelineStatus.toUpperCase() == 'SUCCESS' ?
'green' : 'red'

            def body = """
               <html>
               <body>
               <div style="border: 4px solid ${bannerColor}; padding: 10px;">
               <h2>${jobName} - Build ${buildNumber}</h2>
               <div style="background-color: ${bannerColor}; padding: 10px;">
               <h3 style="color: white;">Pipeline Status:
${pipelineStatus.toUpperCase()}</h3>
               </div>
               <p>Check the <a href="${BUILD_URL}">console output</a>.</p>
               </div>
```

```
                </body>
                </html>
            """

            emailext (
                subject: "${jobName} - Build ${buildNumber} -
${pipelineStatus.toUpperCase()}",
                body: body,
                to: 'jaiswaladi246@gmail.com',
                from: 'jenkins@example.com',
                replyTo: 'jenkins@example.com',
                mimeType: 'text/html',
                attachmentsPattern: 'trivy-image-report.html'
            )
        }
    }
  }
}
```

This is a Jenkins pipeline script written in Groovy syntax. Let's break down each part of the script:

1. **Pipeline Definition**: The pipeline is defined using the `pipeline` block. Inside, it has several sections: `agent`, `tools`, `environment`, `stages`, and `post`.
    - `agent any`: Specifies that the pipeline can run on any available agent.
    - `tools`: Specifies the tools required for the pipeline. In this case, it includes JDK and Maven.
    - `environment`: Defines an environment variable `SCANNER_HOME` which is set to the path of the Sonar Scanner tool.

2. **Stages**: The pipeline defines multiple stages, each representing a different phase of the software delivery process.

    - **Git Checkout**: Checks out the source code from a Git repository.

    - **Compile**: Compiles the source code using Maven.

    - **Test**: Runs the tests using Maven.

    - **File System Scan**: Performs a file system scan using Trivy, a vulnerability scanner.

- **SonarQube Analysis**: Analyzes the code quality using SonarQube.
- **Quality Gate**: Checks the quality gate status from SonarQube.
- **Build**: Packages the application using Maven.
- **Publish To Nexus**: Publishes the built artifacts to a Nexus repository.
- **Build & Tag Docker Image**: Builds and tags a Docker image.
- **Docker Image Scan**: Scans the Docker image for vulnerabilities using Trivy.
- **Push Docker Image**: Pushes the Docker image to a Docker registry.
- **Deploy To Kubernetes**: Deploys the application to a Kubernetes cluster.
- **Verify the Deployment**: Verifies the deployment status by checking pods and services in the Kubernetes namespace.

3. **Post Section**: This section defines actions to be taken after the execution of the pipeline, regardless of its result (success or failure).
    - **Email Notification**: Sends an email notification about the pipeline status. It includes the job name, build number, pipeline status, and a link to the console output. It attaches the Trivy image scan report to the email.
4. **Scripting**: Various parts of the script involve Groovy scripting:
    - Defining variables (`jobName`, `buildNumber`, `pipelineStatus`).
    - Constructing an HTML body for the email notification based on the pipeline status.
    - Using the `emailext` function to send the email notification.
    - Using shell commands (`sh`) to execute commands like Maven, Trivy, Docker, and Kubernetes CLI (`kubectl`).

Overall, this pipeline automates the software delivery process from code checkout to deployment on Kubernetes, including testing, code quality analysis, Docker image creation, vulnerability scanning, and deployment verification, with notifications sent via email.