DevOps Shack

Real-Time Corporate Gitlab CICD Pipeline

[Click Here To Enrol To Batch-5 | DevOps & Cloud DevOps]

Stage 1: Install Required Tools

In this stage, we ensure that all essential tools like JDK, Node.js, and Maven are set up on the CI/CD server.

```
install_tools:
  stage: setup
  script:
    - apt-get update -qy
    - apt-get install -y default-jdk nodejs npm maven
  tags:
    - docker
```

Explanation:

- The stage is named `install_tools` and is part of the `setup` stage.
- The `script` section contains commands to update package lists and install necessary tools using `apt-get`.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 2: Install Project Dependencies

Here, project-specific dependencies are installed using npm or Maven.

```
install_dependencies:
  stage: setup
  script:
    - npm install      # For Node.js projects
```

```
    - mvn clean install   # For Maven projects
  tags:
    - docker
```

Explanation:

- The stage is named `install_dependencies` and is part of the `setup` stage.
- The `script` section contains commands to install project dependencies using npm or Maven depending on the project type.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 3: Execute Test Cases

This stage involves running automated tests to validate the functionality of the code.

```
run_tests:
  stage: test
  script:
    - npm test        # For Node.js projects
    - mvn test        # For Maven projects
  tags:
    - docker
```

Explanation:

- The stage is named `run_tests` and is part of the `test` stage.
- The `script` section contains commands to execute test cases using npm or Maven depending on the project type.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 4: Perform File System Scan (Trivy)

In this stage, project files are scanned for vulnerabilities using Trivy.

```
file_system_scan:
  stage: test
  script:
    - trivy <path_to_project_directory>
  tags:
    - docker
```

Explanation:

- The stage is named `file_system_scan` and is part of the `test` stage.
- The `script` section contains the command to run Trivy with the path to the project directory.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 5: Evaluate Code Quality (SonarQube)

This stage involves evaluating code quality using SonarQube's static analysis.

```
code_quality_analysis:
  stage: test
  script:
    - sonar-scanner
  tags:
    - docker
```

Explanation:

- The stage is named `code_quality_analysis` and is part of the `test` stage.
- The `script` section contains the command to run SonarQube's scanner.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 6: Perform Quality Gate Check

Here, the code is checked against predefined quality standards.

```
quality_gate_check:
  stage: test
  script:
    - mvn sonar:sonar
  tags:
    - docker
```

Explanation:

- The stage is named `quality_gate_check` and is part of the `test` stage.
- The `script` section contains the command to run Maven with SonarQube integration.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 7: Build Application Artifact

This stage involves compiling and packaging the application into an executable artifact.

```
build_artifact:
  stage: build
  script:
    - mvn clean package
  artifacts:
    paths:
      - target/*.jar   # Adjust based on artifact type (e.g., WAR, JAR)
  tags:
    - docker
```

Explanation:

- The stage is named `build_artifact` and is part of the `build` stage.
- The `script` section contains the command to clean and package the application using Maven.
- `artifacts` specify the files to be saved as artifacts for later stages to use.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 8: Publish Artifact to Nexus

In this stage, the artifact is uploaded to Nexus for version control.

```
publish_to_nexus:
  stage: deploy
  script:
    - mvn deploy
  tags:
    - docker
```

Explanation:

- The stage is named `publish_to_nexus` and is part of the `deploy` stage.
- The `script` section contains the command to deploy the artifact using Maven.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 9: Build Docker Image

This stage creates a Docker image for containerized deployment.

```
build_docker_image:
  stage: build
  script:
    - docker build -t your_image_name:latest .
  tags:
    - docker
```

Explanation:

- The stage is named `build_docker_image` and is part of the `build` stage.
- The `script` section contains the command to build a Docker image using the Dockerfile in the project directory.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 10: Scan Docker Image for Security Vulnerabilities (Trivy)

This stage checks the Docker image for security vulnerabilities using Trivy.

```
scan_docker_image:
  stage: test
  script:
    - trivy image your_image_name:latest
  tags:
    - docker
```

Explanation:

- The stage is named `scan_docker_image` and is part of the `test` stage.
- The `script` section contains the command to run Trivy on the Docker image.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 11: Push Docker Image to Docker Hub

This stage uploads the Docker image to Docker Hub for distribution.

```
push_to_docker_hub:
  stage: deploy
  script:
    - docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD
    - docker push your_image_name:latest
  tags:
    - docker
```

Explanation:

- The stage is named `push_to_docker_hub` and is part of the `deploy` stage.
- The `script` section contains commands to log in to Docker Hub using credentials and push the Docker image.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 12: Update Kubernetes Manifests

Here, Kubernetes manifest files are updated to reference the new Docker image.

```
update_kubernetes_manifests:
  stage: deploy
  script:
    - sed -i 's|old_image_name|your_image_name|g' path/to/kubernetes/*.yaml
  tags:
    - docker
```

Explanation:

- The stage is named `update_kubernetes_manifests` and is part of the `deploy` stage.
- The `script` section contains a command to replace the old image name with the new one in Kubernetes manifest files.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 13: Deploy Application to Kubernetes Cluster

This stage deploys the application to a Kubernetes cluster for orchestration.

```
deploy_to_kubernetes:
  stage: deploy
  script:
    - kubectl apply -f path/to/kubernetes/*.yaml
  tags:
    - kubernetes
```

Explanation:

- The stage is named `deploy_to_kubernetes` and is part of the `deploy` stage.
- The `script` section contains a command to apply Kubernetes manifest files to the cluster.
- `tags` specify that this job should run on a runner with the `kubernetes` tag.

Stage 14: Verify Deployment

Here, the successful deployment of the application is confirmed.

```
verify_deployment:
  stage: deploy
  script:
    - kubectl get pods --namespace your_namespace
    # Additional verification steps can be added here
  tags:
    - kubernetes
```

Explanation:

- The stage is named `verify_deployment` and is part of the `deploy` stage.
- The `script` section contains a command to list pods in the namespace to verify deployment.
- `tags` specify that this job should run on a runner with the `kubernetes` tag.

Stage 15: OWASP ZAP Security Testing

This stage conducts security testing with OWASP ZAP to identify vulnerabilities.

```
owasp_zap_security_testing:
  stage: test
  script:
    - zap-cli --url <your_application_url> --spider --ajax --quick-scan --
output <output_file>
  tags:
    - docker
```

Explanation:

- The stage is named `owasp_zap_security_testing` and is part of the `test` stage.
- The `script` section contains a command to run OWASP ZAP CLI to conduct security testing.
- `tags` specify that this job should run on a runner with the `docker` tag.

Stage 16: Send Email Notifications

Notify stakeholders about pipeline status and results via email.

```
send_email_notifications:
  stage: notify
  script:
    - echo "Sending email notifications..."
    # Command to send email notifications using your email service provider
  when: always
  tags:
    - email
```

Explanation:

- The stage is named `send_email_notifications` and is part of the `notify` stage.
- The `script` section contains a command (placeholder) to send email notifications.
- `when: always` ensures that this job runs even if previous stages fail.
- `tags` specify that this job should run on a runner with the `email` tag.

```yaml
stages:
  - setup
  - test
  - build
  - deploy
  - notify

install_required_tools:
  stage: setup
  script:
    - apt-get update -qy
    - apt-get install -y default-jdk nodejs npm maven
  tags:
    - docker

install_project_dependencies:
  stage: setup
  script:
    - npm install     # For Node.js projects
    - mvn clean install   # For Maven projects
  tags:
    - docker

execute_test_cases:
  stage: test
  script:
    - npm test        # For Node.js projects
    - mvn test        # For Maven projects
  tags:
    - docker

perform_file_system_scan:
  stage: test
  script:
    - trivy <path_to_project_directory>
  tags:
    - docker

evaluate_code_quality:
  stage: test
  script:
    - sonar-scanner
  tags:
    - docker

perform_quality_gate_check:
  stage: test
  script:
    - mvn sonar:sonar
  tags:
    - docker

build_application_artifact:
  stage: build
  script:
    - mvn clean package
  artifacts:
    paths:
      - target/*.jar   # Adjust based on artifact type (e.g., WAR, JAR)
  tags:
    - docker
```

```
publish_to_nexus:
  stage: deploy
  script:
    - mvn deploy
  tags:
    - docker

build_docker_image:
  stage: build
  script:
    - docker build -t your_image_name:latest .
  tags:
    - docker

scan_docker_image:
  stage: test
  script:
    - trivy your_image_name:latest
  tags:
    - docker

push_to_docker_hub:
  stage: deploy
  script:
    - docker login -u $DOCKER_USERNAME -p $DOCKER_PASSWORD
    - docker push your_image_name:latest
  tags:
    - docker

update_kubernetes_manifests:
  stage: deploy
  script:
    - sed -i 's|old_image_name|your_image_name|g' path/to/kubernetes/*.yaml
  tags:
    - docker

deploy_to_kubernetes:
  stage: deploy
  script:
    - kubectl apply -f path/to/kubernetes/*.yaml
  tags:
    - kubernetes

verify_deployment:
  stage: deploy
  script:
    - kubectl get pods --namespace your_namespace
    # Additional verification steps can be added here
  tags:
    - kubernetes

owasp_zap_security_testing:
  stage: test
  script:
    - zap-cli --url <your_application_url> --spider --ajax --quick-scan --
output <output_file>
  tags:
    - docker
```

```
send_email_notifications:
  stage: notify
  script:
    - echo "Sending email notifications..."
    # Command to send email notifications using your email service provider
  when: always
  tags:
    - email
```