```
# Use an official Java runtime as a parent image
FROM openjdk:11-jre-slim

# Set environment variables
ENV JMETER_VERSION=5.6.3
ENV JMETER_HOME=/opt/apache-jmeter

# Install dependencies
RUN apt-get update && \
    apt-get install -y wget && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Download and install JMeter
RUN wget -q https://downloads.apache.org/jmeter/binaries/apache-jmeter-$
{JMETER_VERSION}.tgz -O /tmp/jmeter.tgz && \
    mkdir -p ${JMETER_HOME} && \
    tar -xzf /tmp/jmeter.tgz -C ${JMETER_HOME} --strip-components=1 && \
    rm /tmp/jmeter.tgz

# Set JMeter bin directory in PATH
ENV PATH="${JMETER_HOME}/bin:${PATH}"

# Set default command
CMD ["jmeter", "-v"]
```

https://hooks.slack.com/services/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

```
stages:
- check
- download
- unzip
- build_image
# - publish_image
- run_container
- build_images_jmeter
- verify_workspace
- preparation
- run_test
- cleanup

variables:
NEXUS_URL: 'https://nexus.u-cloudsolutions.xyz'
NEXUS_REPOSITORY: 'student-repository'
GROUP_ID: 'com.artificial.GO'
ARTIFACT_ID: 'artifact'
ZIP_FILE_NAME: 'artifact-${COMMIT_ID}.zip'
NEXUS_CREDENTIALS: 'wissem:3FqNnJ6XzF' # Ensure credentials are handled securely
NEXUS_DOCKER_REPOSITORY: 'docker-repository'
IMAGE_NAME: 'my-docker-image'
DOCKER_REGISTRY_URL: 'http://127.0.1.1:10001'
JMETER_HOME: "/opt/apache-jmeter"
JMETER_TEST_FILE: "tests/jmeter/performance-test.jmx"
JMETER_TEST_FILE_TMP: "performance-test.jmx"
JMETER_IMAGE_TAG: "your-jmeter-image:latest"
SLACK_WEBHOOK_URL:
'https://hooks.slack.com/services/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'

# Function to send Slack notifications
.send_slack_notification:
script:
- |
curl -X POST -H 'Content-type: application/json' --data "{\"text\":\"$SLACK_MESSAGE\"}"
$SLACK_WEBHOOK_URL
check_artifact:
stage: check
tags:
- run
script:
- echo "Checking if artifact exists in Nexus..."
- |
MAVEN_GROUP_ID_PATH=${GROUP_ID//./\/}
VERSION_TAG=${COMMIT_ID}
NEXUS_CHECK_URL="${NEXUS_URL}/repository/${NEXUS_REPOSITORY}/$
{MAVEN_GROUP_ID_PATH}/${ARTIFACT_ID}/${VERSION_TAG}/${ZIP_FILE_NAME}"
echo "Artifact URL: ${NEXUS_CHECK_URL}" # Print the URL
RESPONSE_CODE=$(curl -s -o /dev/null -w "%{http_code}" -u "${NEXUS_CREDENTIALS}"
"$NEXUS_CHECK_URL")
```

```
if [ "$RESPONSE_CODE" -eq "200" ]; then
echo "Artifact found in Nexus."
else
echo "Artifact not found. HTTP response code: $RESPONSE_CODE"
exit 1
fi
after_script:
- *send_slack_notification
rules:
- if: '$COMMIT_ID'
when: always
download_artifact:
stage: download
tags:
- run
script:
- echo "Downloading artifact from Nexus..."
- |
mkdir -p artifacts
MAVEN_GROUP_ID_PATH=${GROUP_ID//./\/}
VERSION_TAG=${COMMIT_ID}
NEXUS_DOWNLOAD_URL="${NEXUS_URL}/repository/${NEXUS_REPOSITORY}/$
{MAVEN_GROUP_ID_PATH}/${ARTIFACT_ID}/${VERSION_TAG}/${ARTIFACT_ID}-$
{VERSION_TAG}.zip"
echo "Artifact URL: ${NEXUS_DOWNLOAD_URL}"
curl -v -u "${NEXUS_CREDENTIALS}" -o "artifacts/${ARTIFACT_ID}-${VERSION_TAG}.zip" "$
{NEXUS_DOWNLOAD_URL}"
echo "Downloaded files in artifacts directory:"
ls -l artifacts || echo "No files downloaded."
if [ ! -f "artifacts/${ARTIFACT_ID}-${VERSION_TAG}.zip" ]; then
echo "Error: Artifact not found after download."
exit 1
fi
echo "Artifact downloaded to artifacts/${ARTIFACT_ID}-${VERSION_TAG}.zip"
after_script:
- *send_slack_notification
artifacts:
paths:
- artifacts/${ARTIFACT_ID}-${COMMIT_ID}.zip # Persist the downloaded ZIP file to be used in the next
stage
rules:
- if: '$COMMIT_ID'
when: always



unzip_artifact:
stage: unzip
tags:
```

```
- run
script:
- echo "Unzipping artifact..."
- mkdir -p extracted
- |
if [ ! -f "artifacts/${ARTIFACT_ID}-${COMMIT_ID}.zip" ]; then
echo "Error: Artifact file not found, cannot unzip."
exit 1
fi
- unzip "artifacts/${ARTIFACT_ID}-${COMMIT_ID}.zip" -d extracted/
- echo "Artifact unzipped to extracted/"
after_script:
- *send_slack_notification
dependencies:
- download_artifact # Ensure the job depends on the previous stage's artifacts
rules:
- if: '$COMMIT_ID'
when: always


build_docker_image:
stage: build_image
tags:
- run
script:
- |
echo "Building Docker image..."
docker build -t "${IMAGE_NAME}:${COMMIT_ID}" .
echo "Docker image built: $IMAGE_NAME"
after_script:
- *send_slack_notification
artifacts:
paths:
- image_tag.txt
rules:
- if: '$COMMIT_ID'
when: always


# push_docker_image:
# stage: publish_image
# tags:
# - run
# script:
# - docker_url="${DOCKER_REGISTRY_URL}/repository/${NEXUS_DOCKER_REPOSITORY}" # Correct the
variable assignment
# - echo "Pushing Docker image to Nexus..."
# - echo "3FqNnJ6XzF" | docker login ${docker_url} -u wissem --password-stdin # Use --password-stdin
for security
# - docker tag ${IMAGE_NAME}:${COMMIT_ID} ${docker_url}/${IMAGE_NAME}:${COMMIT_ID}
# - docker push ${docker_url}/${IMAGE_NAME}:${COMMIT_ID}
# - echo "Docker image ${IMAGE_NAME}:${COMMIT_ID} pushed to Nexus repository."
# dependencies:
# - build_docker_image
# rules:
# - if: '$COMMIT_ID'
# when: always

run_docker_container:
stage: run_container
tags:
- run
script:
- |
echo "Running Docker container from image: ${IMAGE_NAME}:${COMMIT_ID}"
CONTAINER_NAME="my-container-${COMMIT_ID}"
HOST_PORT="9094"
CONTAINER_PORT="9090"
echo "Removing any existing container with name: ${CONTAINER_NAME}"
docker rm -f ${CONTAINER_NAME} || true
echo "Running Docker container..."
docker run -d --name ${CONTAINER_NAME} -p ${HOST_PORT}:${CONTAINER_PORT} $
{IMAGE_NAME}:${COMMIT_ID}
after_script:
- *send_slack_notification
rules:
- if: '$COMMIT_ID'
when: always


build_jmeter_image:
stage: build_images_jmeter
tags:
- run
script:
- echo "Building JMeter Docker image"
- docker build -t ${JMETER_IMAGE_TAG} -f dockerfile .
after_script:
- *send_slack_notification
# Stage: Verify Workspace Contents
verify_workspace_contents:
stage: verify_workspace
tags:
- run
script:
- echo "Listing contents of the workspace"
- ls -l ${CI_PROJECT_DIR}
- find ${CI_PROJECT_DIR} -name "performance-test.jmx"
after_script:
- *send_slack_notification
```

```
preparation:
stage: preparation
tags:
- run
script:
- echo "Checking if JMeter test file exists..."
- |
if [ ! -f ${CI_PROJECT_DIR}/${JMETER_TEST_FILE} ]; then
echo "Test file ${JMETER_TEST_FILE} not found in ${CI_PROJECT_DIR}"
exit 1
else
echo "Test file ${JMETER_TEST_FILE} found."
fi
after_script:
- *send_slack_notification

# Stage: Run JMeter Performance Test
run_jmeter_performance_test:
stage: run_test
tags:
- run
script:
- echo "Starting JMeter performance tests"
- |
docker run --rm \
-v ${CI_PROJECT_DIR}:/workspace \
-w /workspace \
${JMETER_IMAGE_TAG} \
sh -c '
echo "Inside Docker: Verifying mounted directory";
mkdir -p /tmp/unzip_dir/tests/jmeter;

# Copy the JMeter test file to the target directory
cp -r /workspace/${JMETER_TEST_FILE} /tmp/unzip_dir/tests/jmeter/${JMETER_TEST_FILE_TMP};

# Run JMeter performance test
jmeter -n -t /tmp/unzip_dir/tests/jmeter/${JMETER_TEST_FILE_TMP} -l /tmp/jmeter-results.jtl;

# Check if results file was created
ls -l /tmp/jmeter-results.jtl;

# Copy results back to the workspace
cp /tmp/jmeter-results.jtl /workspace/jmeter-results.jtl;
'
artifacts:
paths:
- jmeter-results.jtl
when: always
after_script:
```

```
- *send_slack_notification
rules:
- if: '$COMMIT_ID'
when: always


cleanup_docker_container:
stage: cleanup
tags:
- run
script:
- |
echo "Stopping and removing Docker container..."
CONTAINER_NAME="my-container-${COMMIT_ID}"
docker stop ${CONTAINER_NAME} || true
docker rm -f ${CONTAINER_NAME} || true
after_script:
- *send_slack_notification
rules:
- if: '$COMMIT_ID'
when: always
```