

Mid-Term Practical: CI Pipeline with GitHub Actions and Jenkins

Student Name: Nikhil Shankar C S **Course:** PROG8860 - CI/CD

Assignment: Mid-Term Practical

What I'm Building

This project demonstrates a complete CI/CD pipeline for an Android counter application. The goal is to automate the build, test, and deployment process using both GitHub Actions and Jenkins. The final output is a Docker image that serves test reports and lint analysis through a web interface.

Project Structure

```
PROG8860-CICD/
├── Android/                                # Main Android application
│   └── app/
│       ├── src/
│       │   ├── main/
│       │   │   └── java/rebirth/nixaclabs/cicd/
│       │   │       ├── Counter.kt            # Counter logic
│       │   │       └── MainActivity.kt      # UI with Compose
│       └── test/
│           └── java/rebirth/nixaclabs/cicd/
│               └── CounterTest.kt        # Unit tests
└── Dockerfile                                # Multi-stage Docker build
└── build.gradle
.github/
└── workflows/
    └── android-ci.yml                      # GitHub Actions pipeline
Jenkinsfile                                    # Jenkins pipeline
README.md
```

Separate Repository (AndroidReports):

```
└── server.py                                  # Python HTTP server for reports
└── README.md
```

Application Details

What the App Does

Simple counter application with basic functionality:

- Display shows a number (starts at 0)

- "+1" button increases the count
- "-1" button decreases the count
- Built with Kotlin and Jetpack Compose

Test Cases (6 total)

1. Initial counter value is 0
 2. Increment increases count by 1
 3. Decrement decreases count by 1
 4. Multiple increments work correctly
 5. Multiple decrements work correctly
 6. Increment and decrement together work properly
-

How to Build and Run

Local Build

```
cd Android  
./gradlew assembleDebug
```

APK location: [Android/app/build/outputs/apk/debug/app-debug.apk](#)

Run Tests

```
cd Android  
./gradlew test
```

Test reports generated at: [Android/app/build/reports/tests/testDebugUnitTest/index.html](#)

Run Lint

```
cd Android  
./gradlew lint
```

Lint report generated at: [Android/app/build/reports/lint-results-debug.html](#)

Docker Setup

Multi-Stage Build

The Dockerfile uses three stages:

Stage 1 - Builder:

- Installs Android SDK and build tools
- Builds the APK
- Runs unit tests and generates HTML report
- Runs lint analysis and generates HTML report

Stage 2 - Server Repository:

- Uses alpine/git image
- Created a gitrepo for serving reports using python and is available in the below repository.
- Clones the report server from GitHub (<https://github.com/NikhilShankar/AndroidReports.git>)
- Provides server.py for the final stage

Stage 3 - Report Server:

- Uses lightweight Python base image
- Copies test and lint reports from Stage 1
- Copies server.py from Stage 2
- Runs HTTP server on port 9898
- Final image is small (no Android SDK included)

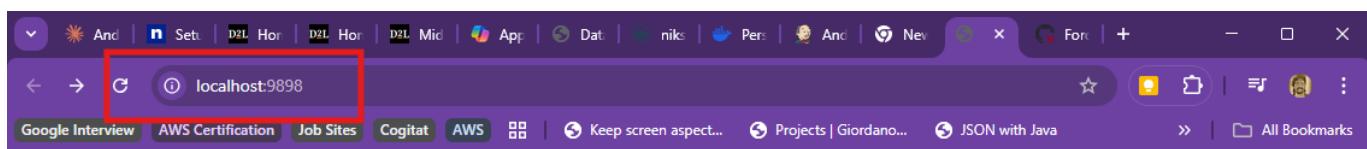
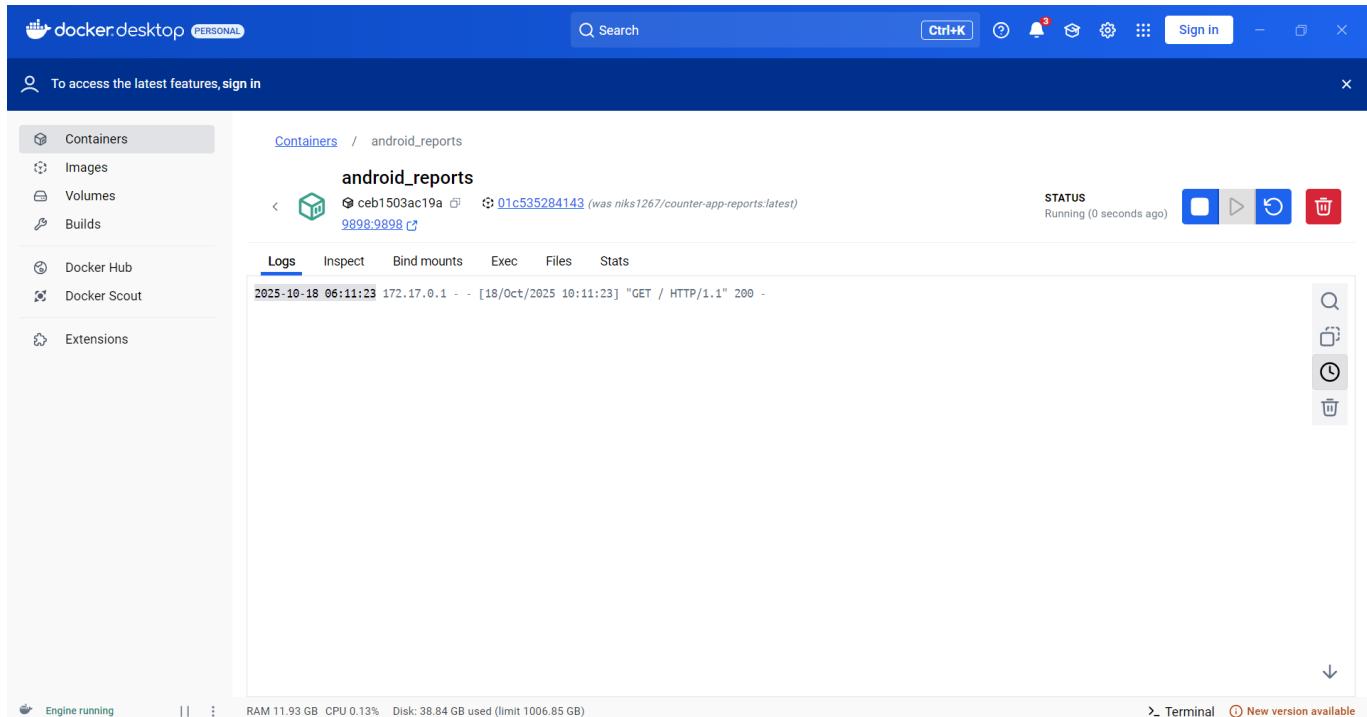
Pull and Run the Image

```
docker pull nikls1267/counter-app-reports:latest
docker run -p 9898:9898 nikls1267/counter-app-reports:latest
```

Access Reports

Open your browser:

- **Home:** <http://localhost:9898>
- **Test Results:** <http://localhost:9898/testresults/index.html>
- **Lint Report:** <http://localhost:9898/lint/index.html>



Android CI Reports

Unit Test Results

Lint Report

localhost:9898/testresults/index.html

Test Summary

6	0	0	0.001s
tests	failures	ignored	duration

100%
successful

Packages **Classes**

Package	Tests	Failures	Ignored	Duration	Success rate
rebirth.nixaclabs.cicd	6	0	0	0.001s	100%

Generated by [Gradle 8.11.1](#) at Oct 18, 2025, 9:40:44 AM

Lint Report: 13 warnings

Check performed at Sat Oct 18 09:41:29 UTC 2025 by AGP (8.9.3)

Overview

Correctness

- 1 ⚠️ [RedundantLabel](#): Redundant label on activity
- 3 ⚠️ [AndroidGradlePluginVersion](#): Obsolete Android Gradle Plugin Version

Performance

- 2 ⚠️ [ObsoleteSdkInt](#): Obsolete SDK_INT Version Check
- 1 ⚠️ [AutoboxingStateCreation](#): State<T> will autobox values assigned to this state. Use a specialized state type instead.
- 7 ⚠️ [UnusedResources](#): Unused resources

Included Additional Checks (60)

Disabled Checks (41)

DISMISS

Redundant label on activity

[./src/main/AndroidManifest.xml](#): 18: Redundant label can be removed

```

15 <activity
16     android:name=".MainActivity"
17     android:exported="true"
18     android:label="@string/app_name"
19     android:theme="@style/Theme.AndroidCICD">
20     <intent-filter>
21         <action android:name="android.intent.action.MAIN" />

```

RedundantLabel Correctness Warning Priority 5/10

GitHub Actions CI Pipeline

Pipeline Stages

The workflow file is located at [.github/workflows/android-ci.yml](#)

What it does:

- Triggers on push to Android folder
- Builds the multi-stage Docker image
- Pushes image to Docker Hub
- All build, test, and lint steps happen inside Docker

How to Test the Pipeline

Push any change to the Android folder:

```
git add .
git commit -m "Update Android app"
git push
```

GitHub Actions automatically runs the pipeline.

11 workflow runs

	Event	Status	Branch	Actor
Updated version in android code base	Android CI #4: Commit 90e1051 pushed by NikhilShankar	master	3 minutes ago	In progress
Merge branch 'master' of github.com:NikhilShankar/PROG8860-CICD	Android CI #3: Commit 2526f57 pushed by NikhilShankar	master	Oct 17, 9:16 PM EDT	4m 4s

github.com/NikhilShankar/PROG8860-CICD/actions/runs/18614204321/job/53076506871

NikhilShankar / PROG8860-CICD

Actions

Summary

Jobs

build-test-and-docker

build-test-and-docker

Run details

Usage

Workflow file

build-test-and-docker

succeeded 47 minutes ago in 5m 48s

Search logs

- Set up job 1s
- Checkout code 1s
- Log in to Docker Hub 1s
- Build Multi-stage Docker Image 5m 40s
- Push Docker Image 4s
- Verify Image 0s
- Post Log in to Docker Hub 0s
- Post Checkout code 0s
- Complete job 0s

Cleaning up orphan processes

Demonstrating Success and Failure

Success scenario: All tests pass (current state)

Failure scenario:

1. Open [Android/app/src/test/java/rebirth/nixaclabs/cicd/CounterTest.kt](#)

2. Change line in `testInitialCountIsZero()`:

```
assertEquals(1, counter.getCount()) // Change 0 to 1
```

3. Push the change

4. Pipeline fails because test fails

5. Revert the change

6. Pipeline passes again

The screenshot shows a Jenkins pipeline interface. On the left, there's a sidebar with 'Summary', 'Jobs' (selected), 'Run details', 'Usage', and 'Workflow file'. The main area is titled 'build-test-and-docker' with a status 'failed now in 5m 16s'. A search bar 'Search logs' and a gear icon are at the top right. The log itself has a header 'Build Multi-stage Docker Image' with a failure icon. The log content is as follows:

```

file:///app/app/build/reports/tests/testDebugUnitTest/index.html
1614 #22 26.52
1615 #22 26.52 * Try:
1616 #22 26.52 > Run with --scan to get full insights.
1617 #22 26.53
1618 #22 26.53 BUILD FAILED in 26s
1619 #22 26.53 39 actionable tasks: 23 executed, 16 up-to-date
1620 #22 ERROR: process "/bin/sh -c ./gradlew test --no-daemon" did not complete successfully: exit
code: 1
1621 -----
1622 > [builder 9/10] RUN ./gradlew test --no-daemon:
1623 26.52
1624 26.52 * What went wrong:
1625 26.52 Execution failed for task ':app:testDebugUnitTest'.
1626 26.52 > There were failing tests. See the report at:
file:///app/app/build/reports/tests/testDebugUnitTest/index.html
1627 26.52
1628 26.52 * Try:
1629 26.52 > Run with --scan to get full insights.
1630 26.53
1631 26.53 BUILD FAILED in 26s
1632 26.53 39 actionable tasks: 23 executed, 16 up-to-date
1633 -----
1634 Dockerfile:36
1635 -----
1636 34 |
1637 35 |     # Run tests and generate test reports
1638 36 | >>> RUN ./gradlew test --no-daemon
1639 37 |
1640 38 |     # Run lint and generate lint reports
1641 -----
1642 ERROR: failed to build: failed to solve: process "/bin/sh -c ./gradlew test --no-daemon" did not
complete successfully: exit code: 1
1643 Error: Process completed with exit code 1.

```

Below the log, there are two steps: 'Push Docker Image' and 'Verify Image', both with a duration of '0s'.

Jenkins Pipeline

Setup

Jenkins runs in Docker:

```
docker run -d \
--name jenkins \
```

```
-p 8080:8080 \
-p 50000:50000 \
-v jenkins_home:/var/jenkins_home \
-v /var/run/docker.sock:/var/run/docker.sock \
jenkins/jenkins:lts
```

Pipeline Configuration

The Jenkinsfile is located at the root of the repository.

Pipeline stages:

1. **Checkout:** Gets code from GitHub
2. **Build & Test:** Builds Docker image (includes tests and lint)
3. **Push to Docker Hub:** Publishes image to registry

Webhook Setup

Jenkins is triggered automatically when code is pushed to GitHub.

Using ngrok for local Jenkins:

1. Install ngrok: <https://ngrok.com/download>
2. Run: `ngrok http 8080`
3. Add webhook to GitHub: <https://your-ngrok-url/github-webhook/>
4. Configure Jenkins job to use "GitHub hook trigger for GITScm polling"

How to Run Jenkins Pipeline

1. Access Jenkins: <http://localhost:8080>
2. Open "Android-Counter-CI" job
3. Click "Build Now"
4. Or push code to GitHub (webhook triggers automatically)

localhost:8080/job/Android-Counter-CI/

Status [Edit description](#)

Changes Build Now Configure Delete Pipeline Stages Rename Pipeline Syntax GitHub Hook Log Credentials

Android-Counter-CI

Android Counter App CI/CD Pipeline

Permalinks

- Last build (#6), 46 min ago
- Last stable build (#5), 56 min ago
- Last successful build (#5), 56 min ago
- Last failed build (#6), 46 min ago
- Last unsuccessful build (#6), 46 min ago
- Last completed build (#6), 46 min ago

Builds

Filter

Build	Result	Time
#6	Failed	10:15 AM
#5	Success	10:05 AM
#4	Success	9:37 AM
#3	Failed	9:34 AM
#2	Failed	2:00 AM
#1	Failed	1:58 AM

REST API Jenkins 2.528.1

Jenkins / Android-Counter-CI / #5 / Console Output

Status Changes Console Output Edit Build Information Delete build '#5' Polling Log Timings Git Build Data Pipeline Overview Replay Pipeline Steps Workspaces Previous Build Next Build

Console Output

Started by GitHub push by NikhilShankar
Obtained Jenkinsfile from git <https://github.com/NikhilShankar/PROG8860-CICD.git>
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/Android-Counter-CI
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository <https://github.com/NikhilShankar/PROG8860-CICD.git>
> git init /var/jenkins_home/workspace/Android-Counter-CI # timeout=10
Fetching upstream changes from <https://github.com/NikhilShankar/PROG8860-CICD.git>
> git --version # timeout=10
> git --version # 'git version 2.47.3'
> git fetch --tags --force --progress -- <https://github.com/NikhilShankar/PROG8860-CICD.git> +refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url <https://github.com/NikhilShankar/PROG8860-CICD.git> # timeout=10
> git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 90e1051ce3b68432bf85d005363fc6acf521f468
(refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 90e1051ce3b68432bf85d005363fc6acf521f468 # timeout=10
Commit message: "Updated version in android code base"
> git rev-list --no-walk e344cb5ac51b4426259a82f40d8e77c30e951b5f # timeout=10
[Pipeline] }
[Pipeline] // stage

Jenkins / Android-Counter-CI / #5

#5 (Oct 18, 2025, 10:05:27 AM)

Status **Add description** **Keep this build forever**

</> Changes **Started by GitHub push by NikhilShankar** Started 58 min ago Took 4 min 3 sec

Console Output **Edit Build Information** **Delete build '#5'** **Polling Log** **Timings** **Git Build Data** **Pipeline Overview** **Restart from Stage** **Replay** **Pipeline Steps** **Workspaces** **Previous Build** **Next Build**

git **Revision:** 90e1051ce3b68432bf85d005363fc6acf521f468 **Repository:** <https://github.com/NikhilShankar/PROG8860-CICD.git>

- refs/remotes/origin/master

</> Changes **1. Updated version in android code base (details / githubweb)**

Changes **REST API** **Jenkins 2.528.1**

github.com/NikhilShankar/PROG8860-CICD/settings/hooks

Google Interview AWS Certification Job Sites Cogitat AWS Keep screen aspect... Projects | Giordano... JSON with Java All Bookmarks

NikhilShankar / PROG8860-CICD Type / to search Settings

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

General Add webhook

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Models

Webhooks

Copilot

Environments

Codespaces

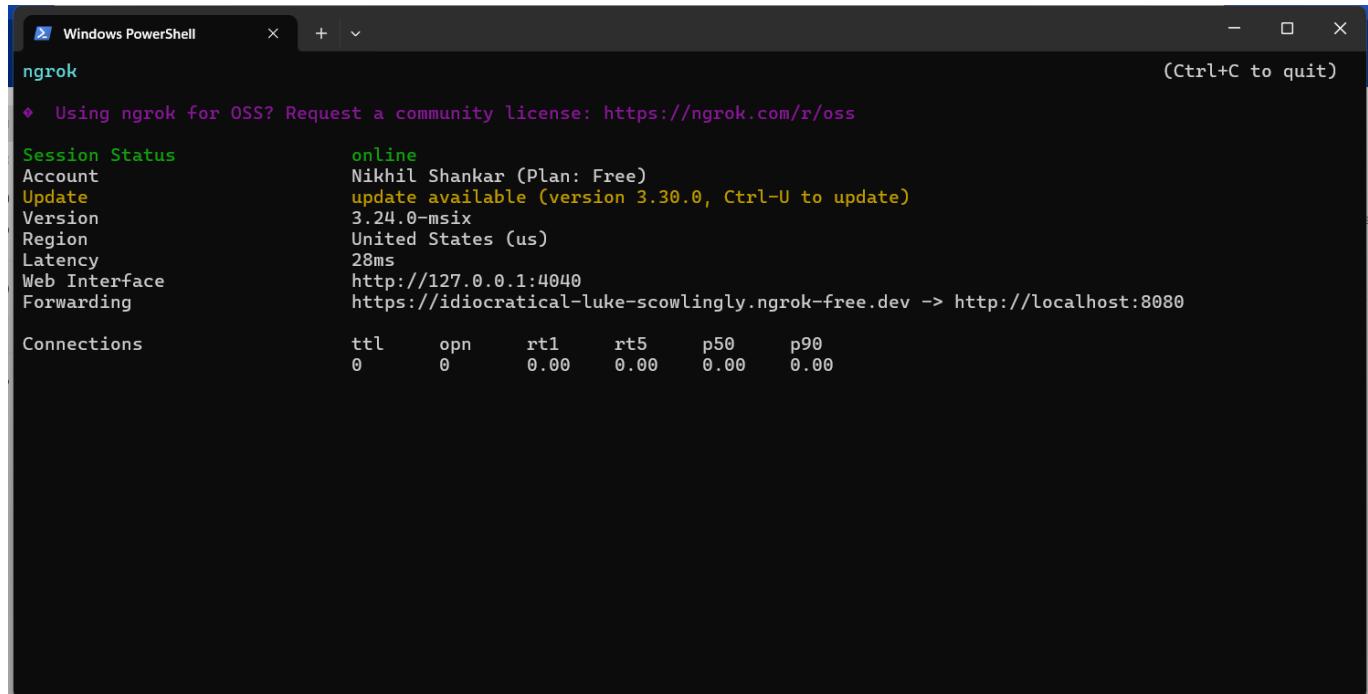
Pages

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

https://idiocritical-luke-scowling.... (push) Edit Delete

Last delivery was successful.



The screenshot shows a Windows PowerShell window titled "Windows PowerShell". The command "ngrok" is run, displaying session details:

```
Using ngrok for OSS? Request a community license: https://ngrok.com/r/oss
Session Status          online
Account                 Nikhil Shankar (Plan: Free)
Update                  update available (version 3.30.0, Ctrl-U to update)
Version                3.24.0-msix
Region                 United States (us)
Latency                28ms
Web Interface          http://127.0.0.1:4040
Forwarding              https://idiocritical-luke-scowlingly.ngrok-free.dev -> http://localhost:8080

Connections            ttl     opn      rt1     rt5     p50     p90
                        0       0       0.00    0.00    0.00    0.00
```

Technologies Used

- **Language:** Kotlin
 - **UI Framework:** Jetpack Compose
 - **Build Tool:** Gradle
 - **Testing:** JUnit
 - **CI/CD:** GitHub Actions, Jenkins
 - **Containerization:** Docker (multi-stage builds)
 - **Report Server:** Python HTTP Server
 - **Registry:** Docker Hub
 - **Webhook Tunneling:** ngrok
-

Key Learnings

Multi-Stage Docker Builds

- Separates build environment from runtime environment
- Reduces final image size significantly
- Clones external repositories during build

CI/CD Automation

- Automated testing catches errors early
- Docker ensures consistent build environment
- Webhooks enable automatic pipeline triggers

Pipeline Design

- Build and test stages run independently
- Reports are generated and served automatically

- Same pipeline works in both GitHub Actions and Jenkins
-

Repository Links

- **Main Repository:** <https://github.com/NikhilShankar/PROG8860-CICD>
 - **Report Server Repository:** <https://github.com/NikhilShankar/AndroidReports>
 - **Docker Hub Image:** <https://hub.docker.com/r/niks1267/counter-app-reports>
-

Conclusion

This project demonstrates a complete CI/CD pipeline with automated testing, Docker containerization, and deployment to a registry. The multi-stage build approach and automated webhooks show real-world CI/CD practices. Both GitHub Actions and Jenkins pipelines successfully build, test, and deploy the application.