
DevOps Pipeline with Jenkins

Submitted By:

Eunjin KIM
s223715707
2025/05/29 10:43

Tutor:

Amin ABKEN

Outcome	Weight
Engage with processes, tools, and practices associated with agile project management across all phases of the dev-ops lifecycle, and use software tools to contribute to real-world projects at each stage in an effective manner	♦♦♦♦◊
Observe and reflect upon the impact of different leadership styles, organisational structures, communication practices, and approaches to conflict management for effective professional relationships within IT companies and projects.	♦♦♦♦◊
Evaluate, review, and synthesise real-world scenarios to inform discussion and practice of IT, and relate to professional practice, codes of ethics, and principles of intellectual property and its protection.	♦♦♦♦◊
Reflect upon professional practice to develop career plans and apply for work opportunities, as ways of engaging in the continuous professional development of discipline-specific and transferable skills.	♦♦◊◊◊

This task is most closely aligned with ULO1 as it involved engaging with multiple DevOps tools (Jenkins, Netlify CLI, SonarQube, GitHub) and practices across the entire pipeline lifecycle. I built a fully automated CI/CD pipeline, incorporating build, test, quality analysis, deployment, release, and monitoring stages, demonstrating hands-on DevOps implementation with real-world tools. It also supports ULO2, as I had to apply effective communication and self-leadership skills to independently

manage this technical project, solve integration issues across services, and document pipeline decisions clearly for assessors. Lastly, the task aligns with ULO3, since I reflected on best practices in secure and ethical code delivery (e.g., interpreting npm audit reports, considering secure deployment, using tools responsibly). This demonstrates awareness of professional standards and real-world

considerations in IT. The task does not relate to ULO4, as it does not involve career planning or job application skills.

May 29, 2025



SIT223/753 – HD Task - Answer Sheet

EUNJIN KIM

223715707

1. A link to the demo video

<https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=c217c2e8-b244-4140-9661-b2e4006917b9>

2. Provide a link to your GitHub repository containing the Jenkins pipeline script. Please ensure that your marking tutor has been granted appropriate access

Git-

<https://github.com/jinyorjin/babynaps>

3. Specify how many stages you have implemented

I have implemented 4 stages in the Jenkins pipeline:

- Build
- Test
- Code Quality
- Deploy
- Security
- Release
- Monitoring

4. A brief description of your project and the technologies used.

BabyNaps Sleep Tracker is a lightweight React web application that allows parents to log their baby's sleep times. It focuses on simplicity and usability, aiming to help with daily tracking.

This project was used to practice CI/CD and DevOps principles using Jenkins.

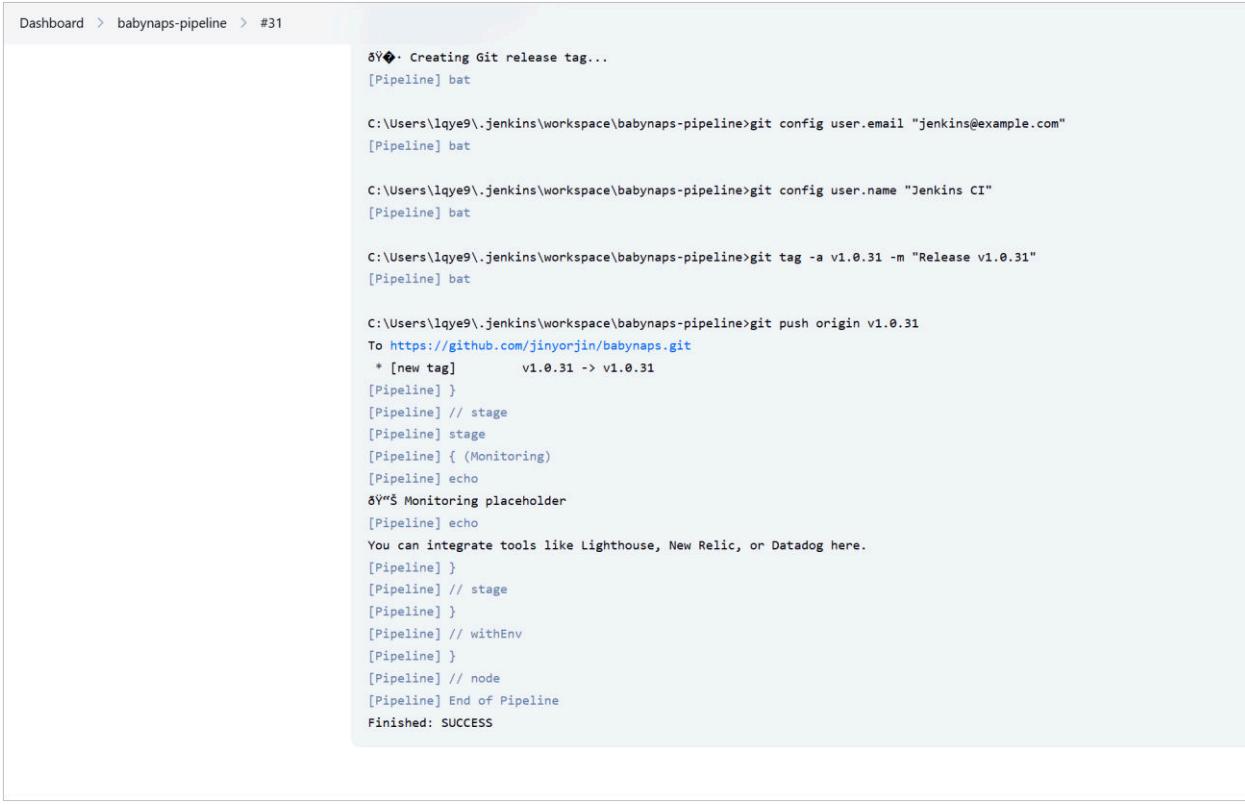
Technologies used:

- **Frontend:** React.js, JavaScript, CSS
- **DevOps / CI/CD:** Jenkins, GitHub, Node.js, SonarQube, sonar-scanner

5. A screenshot of your Jenkins pipeline.

Below is a screenshot of my Jenkins pipeline named babynaps-pipeline. It shows the build history and the latest successful build (#9), which completed all configured stages (Build, Test, SonarQube, Deploy) without errors.

Screenshot attached.



The screenshot shows a Jenkins pipeline build log for the 'babynaps-pipeline' job, specifically build #31. The log output is as follows:

```
Dashboard > babynaps-pipeline > #31
Creating Git release tag...
[Pipeline] bat
C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git config user.email "jenkins@example.com"
[Pipeline] bat
C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git config user.name "Jenkins CI"
[Pipeline] bat
C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git tag -a v1.0.31 -m "Release v1.0.31"
[Pipeline] bat
C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git push origin v1.0.31
To https://github.com/jinyorjin/babynaps.git
 * [new tag]           v1.0.31 -> v1.0.31
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Monitoring)
[Pipeline] echo
$ Monitoring placeholder
[Pipeline] echo
You can integrate tools like Lighthouse, New Relic, or Datadog here.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

 Delete Pipeline

- Last successful build (#31), 15 min ago
- Last failed build (#30), 1 hr 20 min ago
- Last unsuccessful build (#30), 1 hr 20 min ago
- Last completed build (#31), 15 min ago

 GitHub

 SonarQube

 Stages

 Rename

 Pipeline Syntax

 GitHub Hook Log

Builds

...



 Filter /

Today

 #31 3:20 pm



 #30 2:15 pm



```
babynaps-pipeline > #32

  ⚡ Creating Git release tag...
[Pipeline] bat

C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git config user.email "jenkins@example.com"
[Pipeline] bat

C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git config user.name "Jenkins CI"
[Pipeline] bat

C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git tag -a v1.0.32 -m "Release v1.0.32"
[Pipeline] bat

C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git push origin v1.0.32
To https://github.com/jinyorjin/babynaps.git
 * [new tag]           v1.0.32 -> v1.0.32
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Monitoring)
[Pipeline] echo
⚡ Monitoring placeholder
[Pipeline] echo
You can integrate tools like Lighthouse, New Relic, or Datadog here.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

6. A brief description of each stage of your pipeline including the used frameworks/tools.

6.1. Build:

In this stage, Jenkins installs all dependencies using npm install and builds the React app using npm run build.

Tools: Node.js, npm, react-scripts

```
You can control this with the homepage file
```

```
The build folder is ready to be deployed.
```

```
You may serve it with a static server:
```

```
npm install -g serve
```

```
serve -s build
```

```
Find out more about deployment here:
```

```
https://cra.link/deployment
```

```
Run `npm audit` for details.
```

```
[Pipeline] echo
```

```
⇒ Building project...
```

```
[Pipeline] bat
```

```
C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>npm run build
```

```
> babynaps-app@0.1.0 build
```

6.2. Test:

This stage executes any available tests using npm test. Currently, no test files exist, so it completes with code 0.

Tools: Jest (via react-scripts), npm

```
[Pipeline] bat
C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>npm test -- App.test.js --watchAll=false --passWithNoTests

> babynaps-app@0.1.0 test
> react-scripts test App.test.js --watchAll=false --passWithNoTests

No tests found, exiting with code 0
```

6.3. Code Quality

This stage uses SonarQube to check code quality. The Jenkins pipeline runs the sonar-scanner to analyze the project.

As shown in the screenshot:

- No security or maintainability issues
- 2 minor reliability issues
- 0% test coverage (no test files yet)
- No code duplication

Tools: SonarQube, sonar-scanner, Jenkins

```
16:15:05.709 INFO Analysis report generated in 309ms, dir size=259.4 kB
16:15:06.138 INFO Analysis report compressed in 381ms, zip size=47.8 kB
16:15:06.207 INFO Analysis report uploaded in 69ms
16:15:06.212 INFO ANALYSIS SUCCESSFUL, you can find the results at: http://localhost:9000/dashboard?id=babynaps
16:15:06.213 INFO Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
16:15:06.213 INFO More about the report processing at http://localhost:9000/api/ce/task?id=b6f6bd4f-d55b-4e2f-84ae-a702cdce0fb4
```

The screenshot shows the SonarQube interface for the 'BabyNaps Sleep Tracker' project. The main dashboard indicates a successful analysis ('ANALYSIS SUCCESSFUL') with the results available at <http://localhost:9000/dashboard?id=babynaps>. The analysis was completed 14 minutes ago with 237 lines of code and version 1.0. The 'main' branch is shown as 'Passed' with a green checkmark. The 'New Code' section shows 0 new issues, and the 'Accepted issues' section shows 0 valid issues that were not fixed. The overall status is 'Passed'.

The screenshot shows the SonarQube interface at localhost:9000/projects. The main dashboard displays the BabyNaps Sleep Tracker project, which is marked as PUBLIC and has passed its last analysis 14 minutes ago. The project contains 237 Lines of Code in JavaScript and CSS. The quality gate status is green (Passed). The security section shows 0 info issues, 2 low issues, and 0 medium issues. The reliability score is 0.0%, and maintainability is also 0.0%. Coverage and Duplications are both at 0.0%. A sidebar on the left provides filters for Quality Gate (Passed or Failed) and Security (0 info issues, 0 low issues, 0 medium issues).

6.4. Security

This stage runs `npm audit` to scan for known vulnerabilities.

8 issues were detected, including high-severity issues in `nth-check` and `postcss`.

These were not resolved in the current build due to dependency conflicts.

Tools: npm audit, Jenkins

```
C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>npm audit --audit-level=high
# npm audit report
```

SS

6.5. Deployment

This stage is currently a placeholder. It echoes a deploy step for future integration with deployment tools such as **Netlify CLI** or **Docker**.

Tools (planned): Netlify CLI / Docker

This stage uses Netlify CLI to deploy the app to a live production environment.

The app is deployed successfully to: <https://soft-clafoutis-351121.netlify.app>

Tools: Netlify CLI, Jenkins, Node.js

```
[Pipeline] {
[Pipeline] bat

C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>netlify deploy --dir=build --prod --auth=**** --site=****
`- Netlify Build
  `-- Version
    netlify/build_22_1_1
```

6.6. Release

This stage creates a Git tag `v1.0.31` using Jenkins and pushes it to GitHub.

This provides a consistent release versioning strategy.

Tools: git CLI, Jenkins

```
C:\Users\lqye9\.jenkins\workspace\babynaps-pipeline>git push origin v1.0.33
To https://github.com/jinyorjin/babynaps.git
 * [new tag]           v1.0.33 -> v1.0.33
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
```

6.7. Monitoring

This stage is a placeholder, with suggestions to integrate Lighthouse or New Relic in future.

Current monitoring is done via Jenkins build status and logs.

```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Monitoring)
[Pipeline] echo
  ⚡ Monitoring placeholder
[Pipeline] echo
  You can integrate tools like Lighthouse, New Relic, or Datadog here.
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
```

🔗 **Live Site:** <https://soft-clafoutis-351121.netlify.app/>

🔗 **SonarQube Dashboard:** <http://localhost:9000/dashboard?id=babynaps>

Also, please explain how these git commands releases your sample project

```
bat 'git config user.email  
"jenkins@example.com"'  
bat 'git config user.name  
"Jenkins CI"'  
bat 'git tag -a  
v1.0.%BUILD_NUMBER% -m  
"Release  
v1.0.%BUILD_NUMBER%"'  
bat 'git push origin  
v1.0.%BUILD_NUMBER%'
```

-> I used these Git commands in the Release stage to integrate a simple versioning mechanism into the pipeline.

Since each build gets tagged with a version number, it helps maintain a clear history of changes and makes future deployments or troubleshooting more manageable.

I wanted to include this step to practice automating release tracking within a CI/CD workflow.