



Mohammed V University of Rabat
National School of Computer Science and Systems Analysis
Web and Mobile Engineering Department

Data Science and IoT Engineering Program
DevOps Workshop

Automated Testing and Continuous Deployment Pipeline for CI/CD Integration

Prepared by:

OUBALAOUT Siham
RACHED Jihane
ZOUAHRI Jihane
EL MARRHOUB Chaimae
LACHAM Fadwa

Jury :

Pr. MOUMANE Karima

Année Universitaire :2024-2025

Content

INTRODUCTION

OBJECTIVES

TECHNOLOGIES AND TOOLS USED

CHAPTER I. PAINCARE

I.1. PROJECT OVERVIEW

I.2. UNIT TESTING AND CI SETUP

I.3. CONTINUOUS DEPLOYMENT PIPELINE

I.4. CONCLUSION

CHAPTER II. PHP APPLICATION

II.1. PROJECT OVERVIEW

II.2. UNIT TESTING AND CI/CD SETUP

II.3. CONTINUOUS DEPLOYMENT PIPELINE

II.4. CONCLUSION

CONCLUSION

REFERENCES

Introduction

In today's fast-paced software development landscape, Continuous Integration (CI) and Continuous Deployment (CD) have become vital practices for automating key stages of the development process. CI allows developers to frequently integrate their code into a shared repository, where automated tests are run to catch errors early. CD ensures that code changes are automatically deployed to a test or production environment once the CI phase is successful. Together, these practices reduce manual intervention, increase efficiency, and enhance product reliability.

This project aims to demonstrate the setup and integration of CI/CD pipelines for two different applications: one developed in Java and another in PHP. These technologies are commonly used in enterprise and web application development, and the project explores how unit tests can be integrated into CI/CD workflows for both. The focus is on writing automated unit tests and configuring them to run automatically during the CI phase, followed by continuous deployment if the tests pass. For the Java application, tools such as JUnit for testing and Jenkins for CI/CD orchestration were used. The PHP project involved PHPUnit for testing, along with similar CI/CD tools. Both projects faced distinct challenges in terms of configuration and deployment, providing valuable insights into managing CI/CD pipelines across different technology stacks. The end goal was to create fully automated deployment pipelines that reduce errors and speed up delivery.

This report is structured to first introduce the technologies and tools used in both the Java and PHP projects. It then details the process of writing and integrating unit tests into the CI/CD pipelines for each application, followed by a description of the continuous deployment setup. The final sections reflect on the overall project experience, highlighting key learnings and challenges encountered while building these pipelines.

Objectives

Implement Automated Unit Testing:

- Write unit tests for both the **Java** and **PHP** applications to ensure code quality and reliability.
- Configure these tests to automatically run during the **Continuous Integration (CI)** phase, immediately after new code is committed to the repository.

Set Up a Continuous Integration (CI) Pipeline:

- Create a fully automated CI pipeline using tools like **Jenkins**, run tests, and catch errors early in the development cycle for both Java and PHP applications.

Create a Continuous Deployment (CD) Pipeline:

- Build a **Continuous Deployment (CD)** pipeline that automatically deploys the applications to test or production environments once all unit tests pass successfully.
- Ensure that the deployment process is seamless, requiring minimal manual intervention.

Ensure Cross-Technology Compatibility:

- Address the challenges of setting up CI/CD pipelines for different technology stacks (Java and PHP), and ensure that both applications are deployed efficiently and correctly to their respective environments.

Improve Development Workflow:

- Demonstrate how automating testing and deployment can streamline the development process, reduce manual errors, and shorten release cycles.

Technologies and tools used

1 - Testing Frameworks:

- **JUnit**: A testing framework for writing and running unit tests in the Java application.
- **PHPUnit**: A testing framework for writing and running unit tests in the PHP application.

2 - Continuous Integration/Continuous Deployment (CI/CD) Tools:

- **Jenkins** : to automate building, testing, and deploying our app.
- **ngrok** : helped us expose our local web app online for testing and webhook integration.
- **Docker** : to run our app in isolated containers for consistent environments.
- **Git/GitHub** : for version control to track changes and work with Jenkins.

3-Installing Jenkins plugins:

- Git
- Docker
- Docker pipeline
- PHP

Chapitre I. Java App

I.1. Introduction :

When developing applications, it's essential to identify key features because our tests are based on these functionalities

I.2. App overview :

Key Features

PainCare provides several essential features to support its users:

1. **Authentication:** Users can create secure accounts and log in to access personalized features, ensuring that their data remains private and protected.
2. **Symptom Tracking:** The application includes a dedicated menstrual tracking form, allowing users to record details about their symptoms, such as pain intensity and location. This data is transformed into dynamic graphs and statistics, enabling users to visualize their symptom progression over time.
3. **Community Blog:** PainCare fosters a supportive community by allowing users to share articles and personal experiences. The blog serves as a platform for discussion, information exchange, and mutual support among members.
4. **Endometriosis Information:** A dedicated section provides reliable resources and links to relevant articles, enhancing users' understanding of endometriosis and promoting informed health decisions.
5. **Personalized Diagnostics:** Users can generate customized diagnostics based on their symptom tracking data. This feature includes specific recommendations and guidance tailored to individual needs.
6. **Results Downloading:** PainCare enables users to download their personalized diagnostic results and symptom tracking data, facilitating easier sharing with healthcare teams for informed discussions.
7. **Alerts and Reminders:** The application includes customizable alerts to keep users informed about appointments, symptom tracking, community activities, and available results, ensuring they stay engaged with their health management.

I.3. Unit testing and CI/CD Setup

1.2. Example of Unit Testing with JUnit and Mockito

The following example demonstrates how to implement unit tests for a DAO (Data Access Object) class responsible for managing alarms within the PainCare application.

```
@Test
public void testInsertAlarm() throws SQLException {
    // Define the alarm object and set up the mock behavior for inserting an alarm
    Alarm alarm = new Alarm();
    alarm.setTitle("Test Alarm");
    alarm.setTime(Time.valueOf("12:00:00"));
    alarm.setRepeatDays(Set.of("Monday", "Tuesday")); // Ensure this is non-null

    // Execute the DAO method
    alarmDAO.insertAlarm(alarm, 1);

    // Verify that the insert operation was called
    verify(mockPreparedStatement, times(1)).executeUpdate();

    // Verify that the alarm ID was set correctly from the generated keys
    assertEquals(1, alarm.getAlarmId()); // Check if the ID is set correctly
}
```

Test Method: The `testInsertAlarm` method tests the `insertAlarm` functionality of the `AlarmDAOImpl` class. We create an `Alarm` object and execute the insertion method.

Verifications: After executing the method, we verify that the `executeUpdate` method was called exactly once and check that the alarm ID was set correctly based on the mocked `ResultSet`.

1.3. Maven Test Execution

In addition to implementing unit tests, we utilized Maven as our build automation tool to manage the testing process efficiently. Below is a screenshot demonstrating the successful execution of our unit tests within the Maven environment.

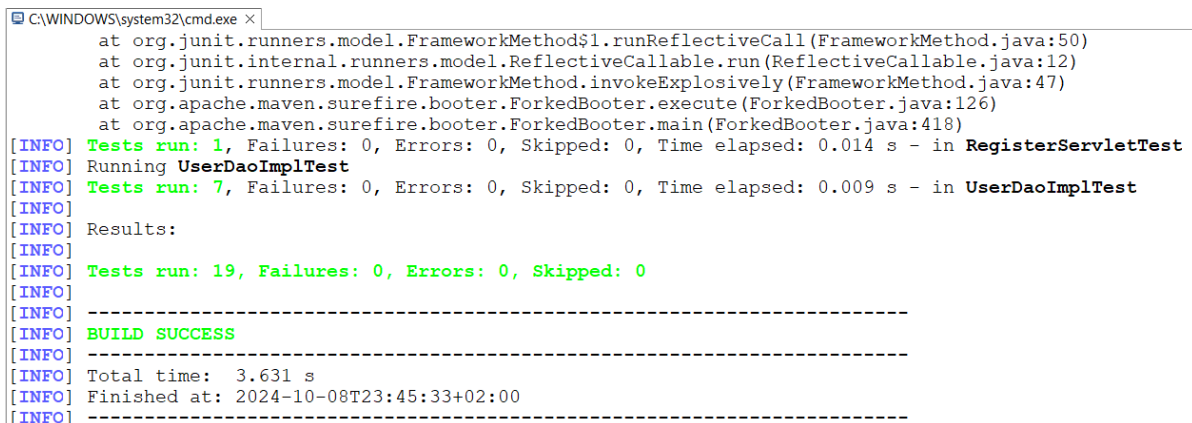
Running Tests Locally with Maven

To run the tests locally, we used the following command in the terminal:

```
C:\Users\pc\eclipse-workspace\PainCare> mvn test
```

This command triggers the execution of all unit tests within the project, providing feedback on the success or failure of each test case.

Test Success Screenshot



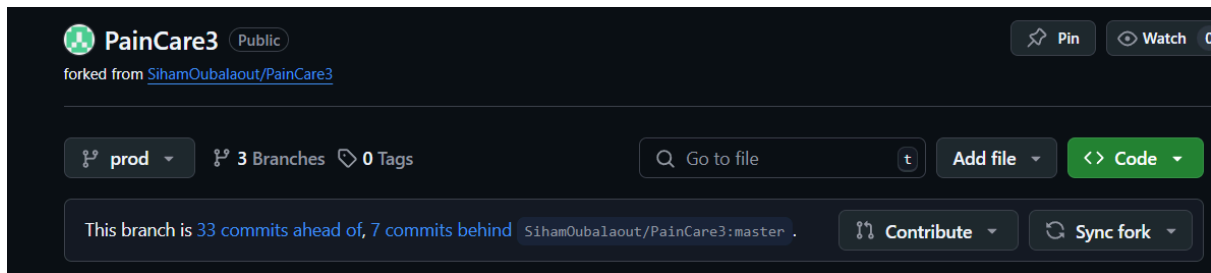
```
C:\WINDOWS\system32\cmd.exe x
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)
    at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
    at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)
    at org.apache.maven.surefire.booter.ForkedBooter.execute(ForkedBooter.java:126)
    at org.apache.maven.surefire.booter.ForkedBooter.main(ForkedBooter.java:418)
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.014 s - in RegisterServletTest
[INFO] Running UserDaoImplTest
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.009 s - in UserDaoImplTest
[INFO] Results:
[INFO] Tests run: 19, Failures: 0, Errors: 0, Skipped: 0
[INFO] BUILD SUCCESS
[INFO] Total time: 3.631 s
[INFO] Finished at: 2024-10-08T23:45:33+02:00
```

In the screenshot, you can observe the results of the test run, including:

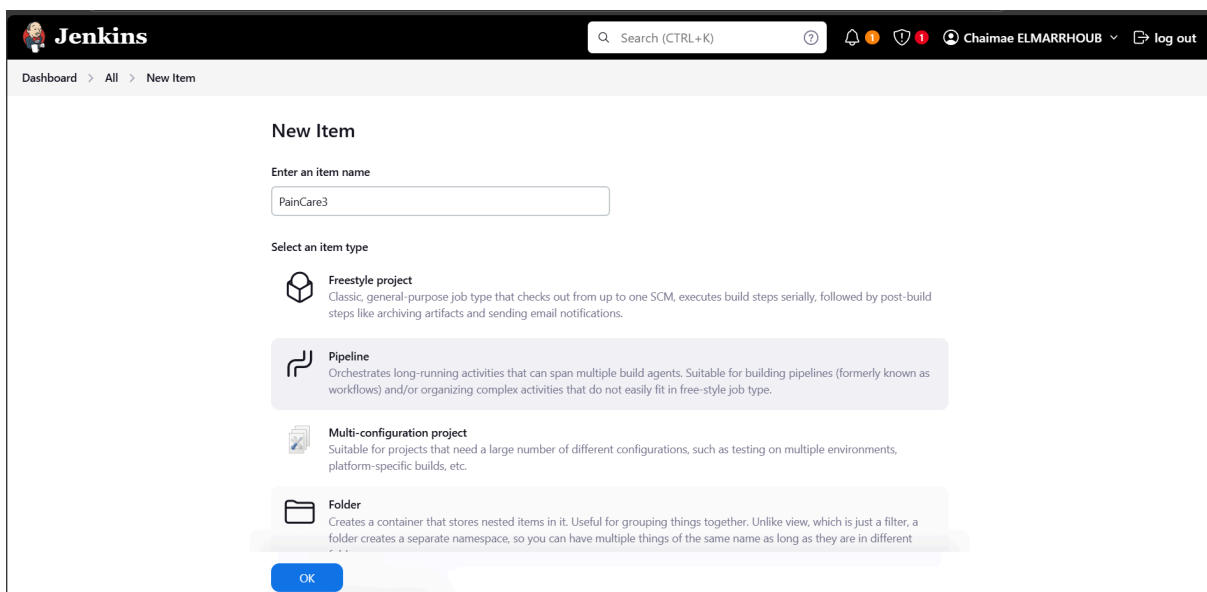
- The number of tests executed.
- The number of successful tests.
- Any failed tests, along with the details for debugging purposes.

2. Setup CI/CD:

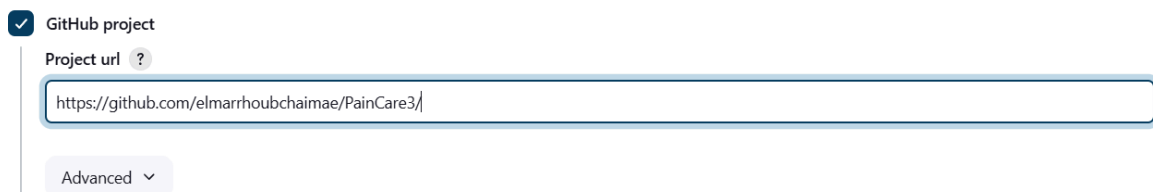
- Fork the GitHub repository as public: <https://github.com/SihamOubalaout/PainCare3>



- Create new branch prod
- Create an admin user or Skip and continue as Admin in Jenkins or login if you already have an account.
- Save and finish the setup.
- Create a new Pipeline item in Jenkins:



- Configure it as a GitHub project and provide your repository URL



- Check the github trigger that we will create later

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ GitHub hook trigger for GITScm polling ?

- For the Pipeline Definition, choose "Script from SCM".

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/elmarhoubchaimae/PainCare3

Credentials ?

token /***** (tokens)

- Make sure to choose the branch “prod” and verify jenkinsFile nomination

Branch Specifier (blank for 'any') ?

*/prod

Add Branch

Repository browser ?

(Auto)

Additional Behaviours

Add

Script Path ?

JenkinsFile

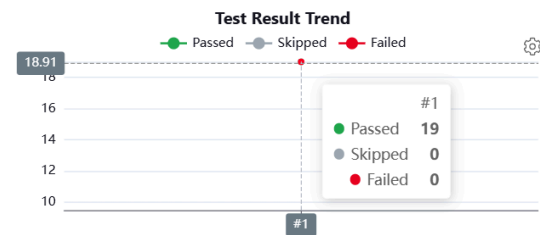
- Try building the project manually

✓ Testing

 Latest Test Result (no failures)

Permalinks

- Last build (#1), 1 min 8 sec ago
- Last stable build (#1), 1 min 8 sec ago
- Last successful build (#1), 1 min 8 sec ago
- Last completed build (#1), 1 min 8 sec ago



- To create a public URL for our webhook, we used "ngrok" to generate a secure public URL that redirects to our local server.
- sign up to <https://ngrok.com/>
- get Your Authtoken from dashboard:

Your Authtoken

This is your personal Authtoken. Use this to authenticate the ngrok agent that you downloaded.

.....
Copy

- Install ngrok for desktop
- Unzip the file and open ngrok in terminal
- Run on terminal : ngrok authtoken <your_authtoken>

```
C:\Users\elmar\Downloads\ngrok-v3-stable-windows-amd64>ngrok authtoken 2nCspEAbBKKk1LOzfxord8J7Eaks_4rk7pxmUHQvMQvmyhKymw
Authtoken saved to configuration file: C:\Users\elmar\AppData\Local\ngrok\ngrok.yml
```

- Run : ngrok http <port>

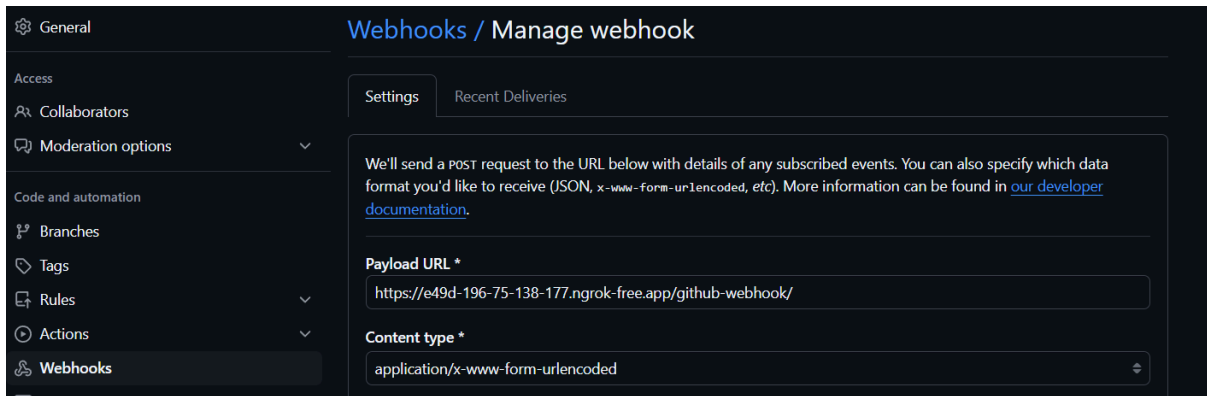
```
C:\Users\elmar\Downloads\ngrok-v3-stable-windows-amd64>ngrok http 9090
```

```
ngrok
Found a bug? Let us know: https://github.com/ngrok/ngrok

Session Status      online
Account             elmarhoub.chaimae@gmail.com (Plan: Free)
Version             3.16.1
Region              Europe (eu)
Web Interface        http://127.0.0.1:4040
Forwarding           https://b515-196-75-205-83.ngrok-free.app -> http://localhost:9090

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

- Copy the URL provided example : <https://df20-196-75-138-177.ngrok-free.app>
- Set up Your webhook to: <https://df20-196-75-138-177.ngrok-free.app/github-webhook/> in your repo settings.



- Clone your repository on your local machine.

git clone <your repo>

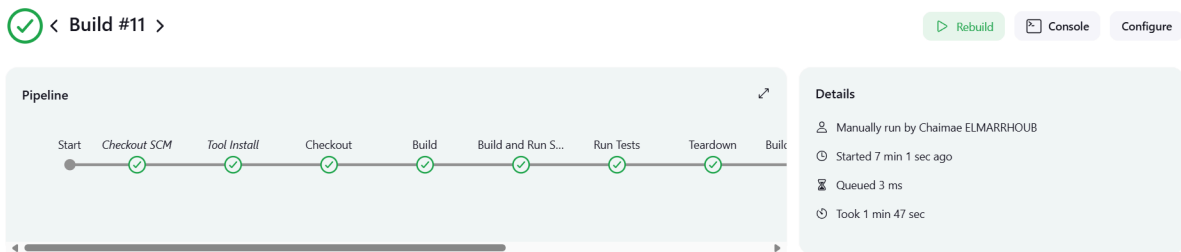
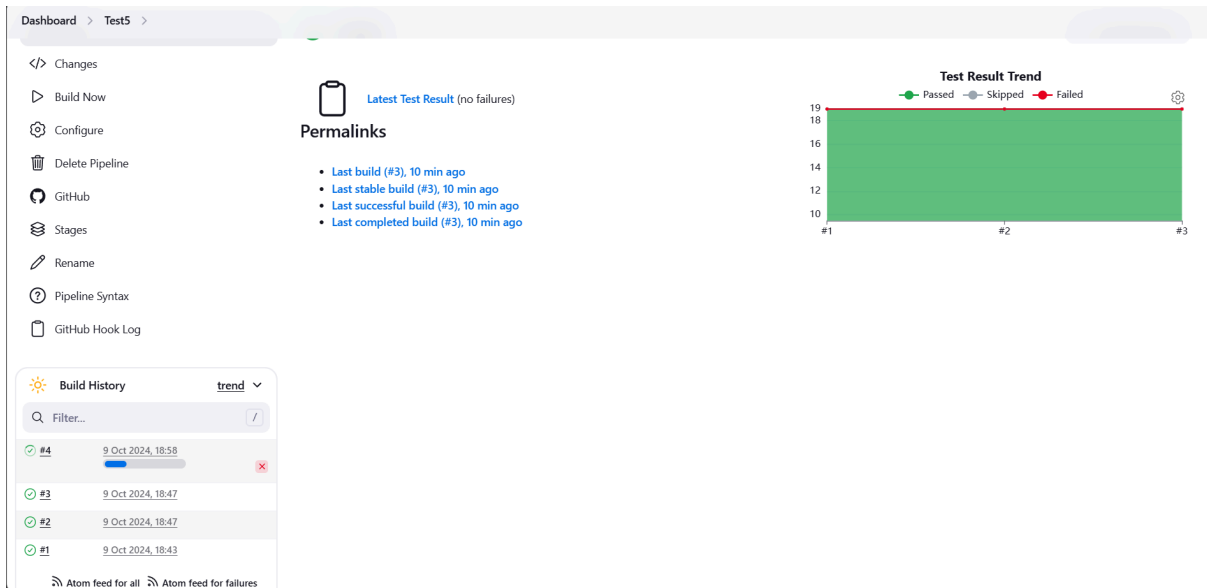
```
C:\Users\elmar\Desktop\testJEE>git clone https://github.com/elmarhoubchaimae/PainCareJEE.git
Cloning into 'PainCareJEE'...
remote: Enumerating objects: 411, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (19/19), done.
remote: Total 411 (delta 10), reused 6 (delta 6), pack-reused 386 (from 1)
Receiving objects: 100% (411/411), 5.27 MiB | 10.53 MiB/s
Receiving objects: 100% (411/411), 10.76 MiB | 13.29 MiB/s, done.
Resolving deltas: 100% (83/83), done.
```

- docker build -t jee-app-tomcat .
- docker run -p 8081:8080 jee-app-tomcat

<input type="checkbox"/>	mysql 4bc6bc963e6d	5.7	In use	10 months ago	688.74 MB	▶	:	🗑
<input type="checkbox"/>	jee-app-tomcat 2fba4ea3a46b	latest	In use	8 minutes ago	1.06 GB	▶	:	🗑

- Switch to the "prod" branch, apply updates to the source file, and push.

```
PS C:\Users\elmar\Desktop\testJEE> cd PainCareJEE
PS C:\Users\elmar\Desktop\testJEE\PainCareJEE> git checkout prod
Switched to a new branch 'prod'
branch 'prod' set up to track 'origin/prod'.
PS C:\Users\elmar\Desktop\testJEE\PainCareJEE> git add .
PS C:\Users\elmar\Desktop\testJEE\PainCareJEE> git commit -m "modit"
[prod 852d922] modit
1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\elmar\Desktop\testJEE\PainCareJEE> git push origin prod
```

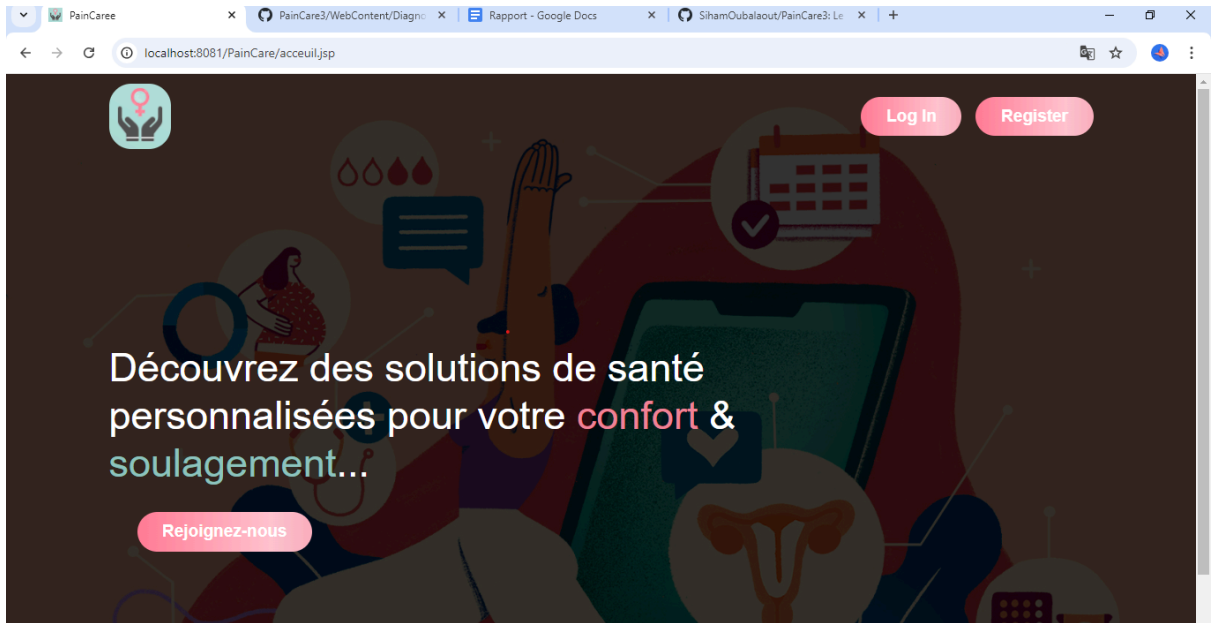


- The CD pipeline has completed successfully, and now you can access the application using the URL configured in the Jenkinsfile, which is running on the Tomcat server.

localhost:8080/job/Paincare1/78/console

Dashboard > Paincare1 > #78

```
[Pipeline] {
[Pipeline] echo
Deployment complete. Application should be running at http://localhost:8081/PainCare/accueil.jsp
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] // withEnv
[Pipeline] // withEnv
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```



Chapitre II. Laravel App

1. Fork Repository

Fork the app from the Git repository: <https://github.com/ZJihane/My-Attendance>.

2. Clone Repository

clone the forked repository

```
PS C:\Users\HP\Desktop\Demo 2> git init
Initialized empty Git repository in C:/Users/HP/Desktop/Demo 2/.git/
PS C:\Users\HP\Desktop\Demo 2> git clone https://github.com/jiihane/My-Attendance.git
Cloning into 'My-Attendance'...
remote: Enumerating objects: 626, done.
remote: Counting objects: 100% (39/39), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 626 (delta 28), reused 31 (delta 22), pack-reused 587 (from 1)
Receiving objects: 100% (626/626), 377.81 KiB | 392.00 KiB/s, done.
Resolving deltas: 100% (313/313), done.
PS C:\Users\HP\Desktop\Demo 2>
```

3. Run composer install

```
PS C:\Users\HP\Desktop\Demo 2> cd .\My-Attendance\
PS C:\Users\HP\Desktop\Demo 2\My-Attendance> composer install
Installing dependencies from lock file (including require-dev)
Verifying lock file contents can be installed on current platform.
Package operations: 128 installs, 0 updates, 0 removals
Information : impossible de trouver des fichiers pour le(s) modèle(s) spécifié(s).
Information : impossible de trouver des fichiers pour le(s) modèle(s) spécifié(s).
- Installing dasprid/enum (1.0.6): Extracting archive
- Installing bacon/bacon-qr-code (2.0.8): Extracting archive
- Installing voku/portable-ascii (2.0.1): Extracting archive
- Installing symfony/polyfill-php80 (v1.31.0): Extracting archive
```

4. create file .env

5. copy content of .env.example in file .env

6. create a database with the the provided name in the .env file

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=my-attendance
DB_USERNAME=root
DB_PASSWORD=
```

7. generate key

```
PS C:\Users\HP\Desktop\Demo 2\My-Attendance> php artisan key:generate
INFO Application key set successfully.
PS C:\Users\HP\Desktop\Demo 2\My-Attendance>
```

8. php artisan migrate

```
PS C:\Users\HP\Desktop\Demo 2\My-Attendance> php artisan migrate

INFO: Preparing database.

Creating migration table ..... 14ms DONE

INFO: Running migrations.

2014_10_12_000000_create_users_table ..... 30ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 34ms DONE
2014_10_12_200000_add_two_factor_columns_to_users_table ..... 11ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 40ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 50ms DONE
2023_04_17_023753_create_parents_table ..... 10ms DONE
2023_04_17_023825_create_classes_table ..... 11ms DONE
2023_04_17_023836_create_eleves_table ..... 124ms DONE
2023_04_17_023918_create_enseignants_table ..... 16ms DONE
2023_04_17_023937_create_matières_table ..... 98ms DONE
2023_04_17_023938_create_seances_table ..... 57ms DONE
2023_04_17_024002_create_presences_table ..... 113ms DONE
2024_10_06_175907_create_sessions_table ..... 60ms DONE

PS C:\Users\HP\Desktop\Demo 2\My-Attendance>
```

9. npm install vite

```
PS C:\Users\HP\Desktop\Demo 2\My-Attendance> npm install vite --save-dev
```

10. run npm dev

```
PS C:\Users\HP\Desktop\Demo 2\My-Attendance> npm run dev

> dev
> vite

VITE v4.5.5 ready in 776 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h to show help

LARAVEL v9.52.16 plugin v0.7.8

→ APP_URL: http://localhost
```

11. testing locally

```
PS C:\Users\HP\Desktop\Demo 2\My-Attendance> php artisan test

Warning: TTY mode is not supported on Windows platform.

PASS Tests\Unit\ExampleTest
✓ that true is true

PASS Tests\Feature\AuthenticationTest
✓ login screen can be rendered
✓ user can login with valid credentials
✓ user cannot login with invalid credentials
✓ user can logout
✓ user cannot login with unverified email

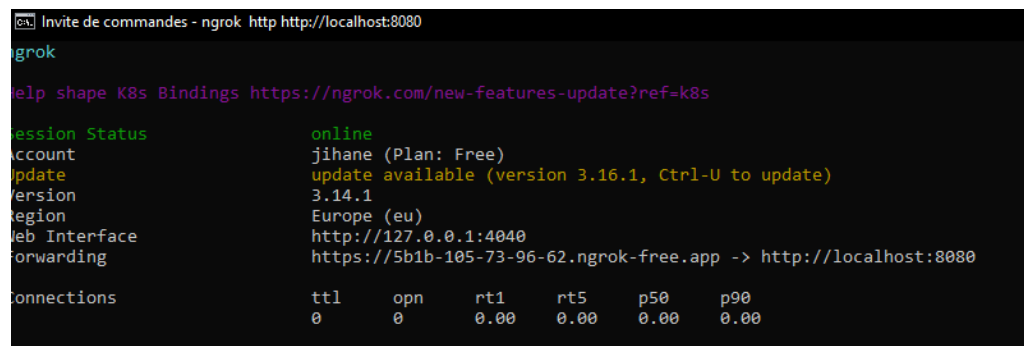
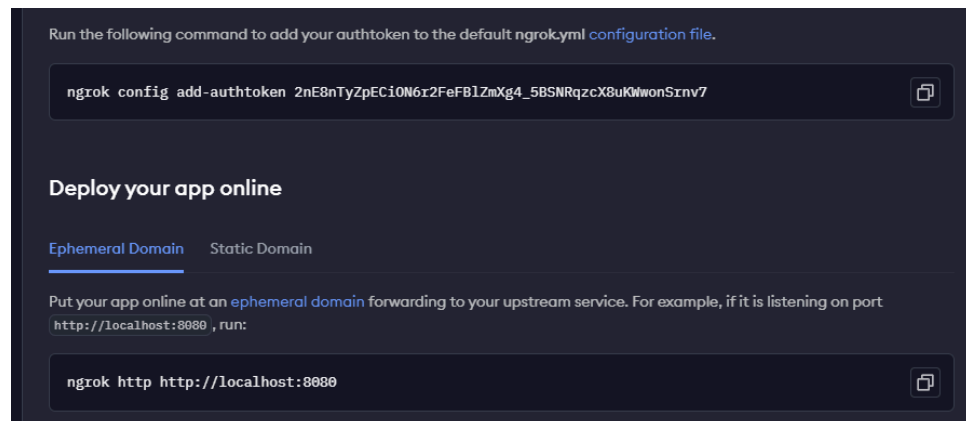
PASS Tests\Feature\ExampleTest
✓ the application returns a successful response

PASS Tests\Feature\RegistrationTest
✓ registration screen can be rendered
✓ user can register with valid credentials
✓ user cannot register with invalid data
✓ user cannot register with existing email
✓ registration fails if password confirmation does not match

Tests: 12 passed
Time: 2.72s
```

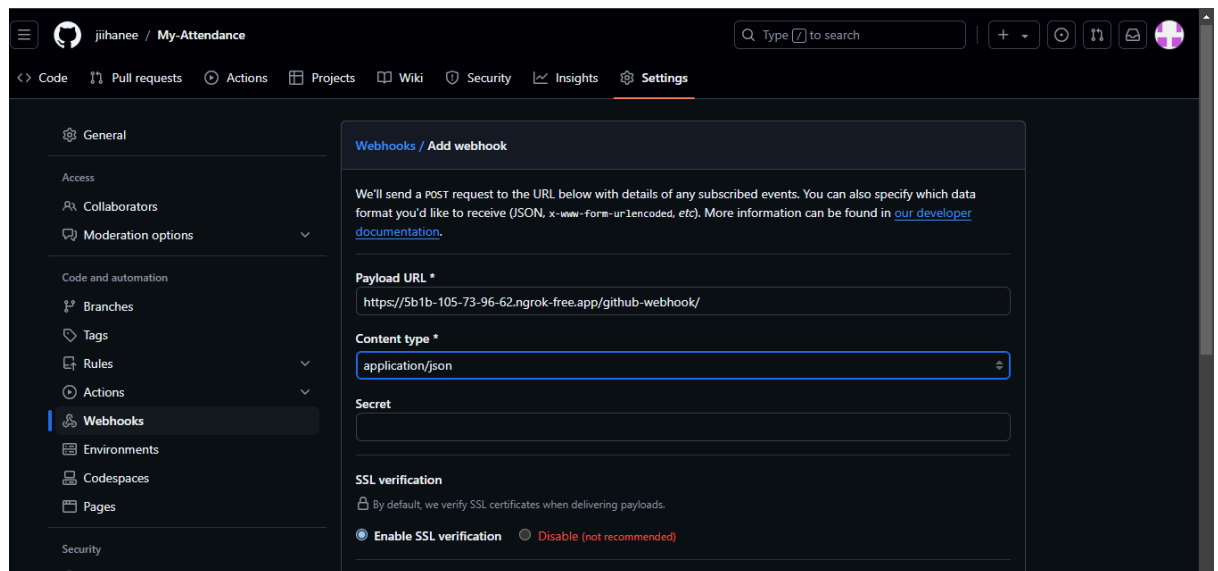
12. ngrok

- install ngrok if not installed
- run in cmd line :



13. web hook

add the webhook on the forked repo



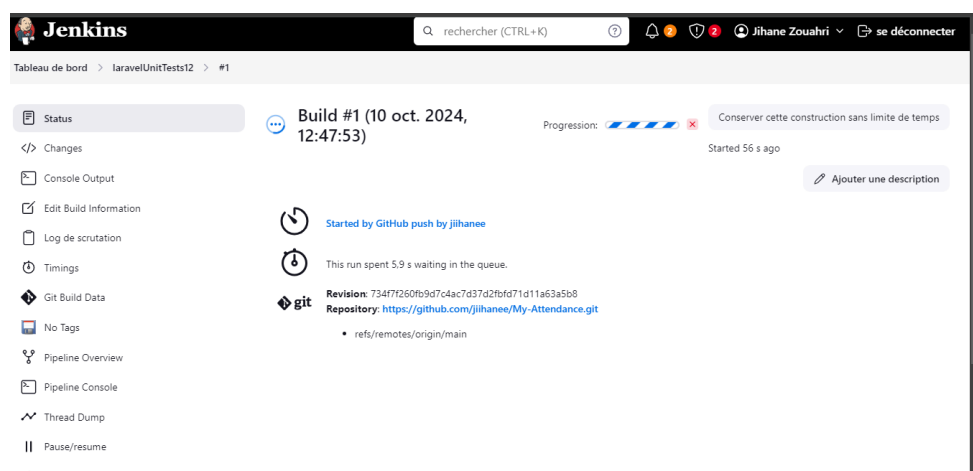
14. create pipeline

15. try pushing

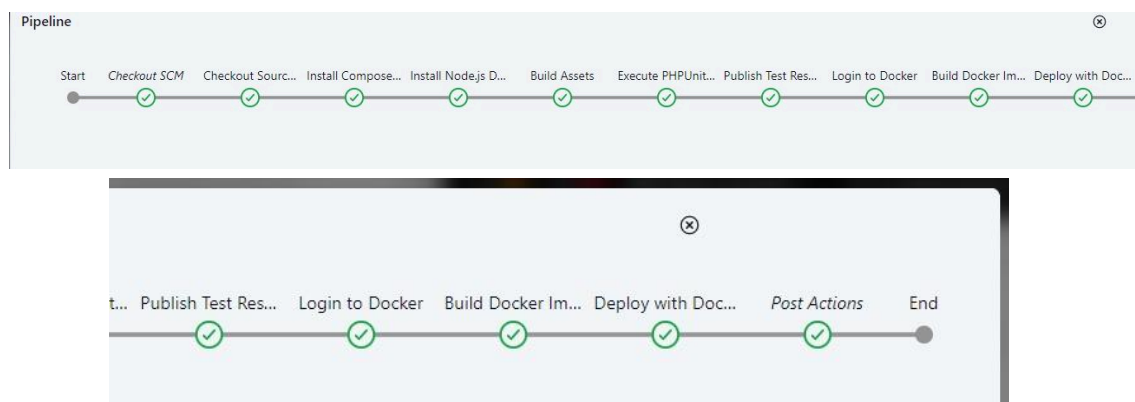
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Users\HP\Desktop\Demo 2\My-Attendance> git add *
● PS C:\Users\HP\Desktop\Demo 2\My-Attendance> git commit -m "test"
[main 734f7f2] test
3 files changed, 4 insertions(+), 4 deletions(-)
● PS C:\Users\HP\Desktop\Demo 2\My-Attendance> git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 537 bytes | 268.00 KiB/s, done.
Total 7 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 4 local objects.
To https://github.com/jiihane/My-Attendance.git
4a96489..734f7f2 main -> main
● PS C:\Users\HP\Desktop\Demo 2\My-Attendance> 
```

```
12:40:06.257 +10 POST /github-webhook/ 200 OK
```



16. The CD pipeline, if successfully completed, should resemble this one :

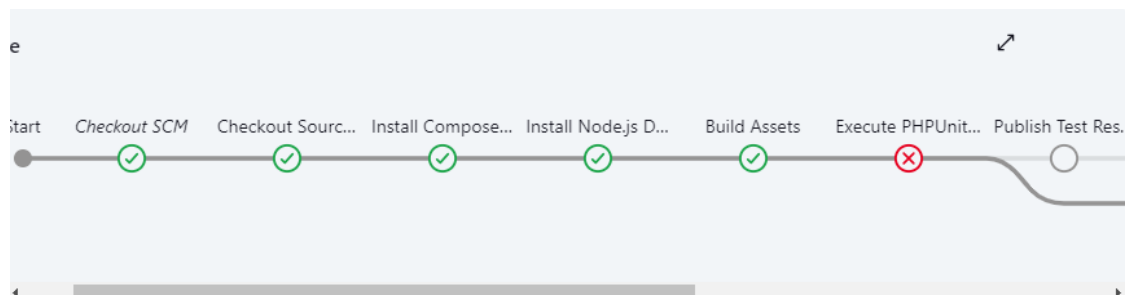


17. try to fail a test in purpose

```

My-Attendance > tests > Unit > ExampleTest.php
1  <?php
2
3  namespace Tests\Unit;
4
5  use PHPUnit\Framework\TestCase;
6
7  class ExampleTest extends TestCase
8  {
9      /**
10       * A basic test example.
11       */
12     public function test_that_true_is_true(): void
13     {
14         // This will fail because false is not true.
15         $this->assertTrue(false);
16     }
17 }
18

```



Conclusion

This guide explained how to set up a CI/CD pipeline for both Java and PHP applications using Jenkins for automating the build, testing, and deployment processes, Docker for ensuring consistent environments across development and production, and GitHub for version control and seamless integration with Jenkins.

