

A thick dark blue vertical bar is positioned on the left side of the page. A blue arrow points to the right from this bar, containing the date. Below the bar, several thin, curved lines in shades of blue and grey sweep upwards and to the right.

27/02/2024

TP DevOps

Utilisation de Jenkins, Maven &
SonarQube

Amara Konte

B3 CYBERSECURITE - YNOV

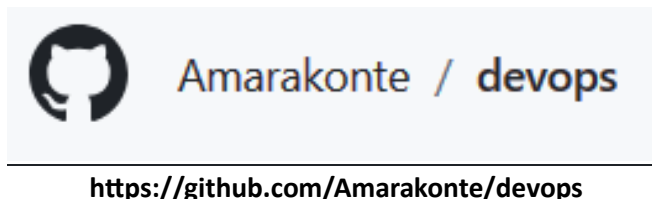
SOMMAIRE

| | |
|--|----|
| 1. GIT | 2 |
| 1.1. Créer un repo GIT Ajouter les branches "prod" et "dev" Charger sur la branche "dev" les fichiers du projet démo..... | 2 |
| 2. Compilation..... | 3 |
| 2.1. Créer un job Jenkins qui récupère le projet sur le repo GIT et démarre une compilation via Maven..... | 3 |
| 2.2. Vérifier que le projet compile..... | 6 |
| 2.3. Ne lance pas de tests | 6 |
| 2.4. Publie la JavaDoc..... | 6 |
| 2.5. Se lance toute les 3 min si changement du SCM | 7 |
| 3. Tests Maven | 7 |
| 3.1. Ajouter JUnit pour faire des tests dans le projet pour créer plusieurs tests et quelques méthodes simples de calcul. | 7 |
| 3.2. Integer add(Integer a, Integer b)..... | 7 |
| 3.3. Integer multi(Integer a, Integer b)..... | 7 |
| 3.4. Integer div(Integer a, Integer b) | 7 |
| 3.5. Dans un nouveau job Jenkins :..... | 8 |
| 3.5.1. Vérifier que la compilation et les tests sont passants. | 8 |
| 3.5.2. Modifier un test de manière à la faire échouer..... | 8 |
| 3.5.3. Vérifier que la compilation fonctionne et que le test remonte en KO..... | 9 |
| 3.5.4. Ignorer le test via Maven de façon à obtenir un build correct. | 9 |
| 3.5.5. Publier le rapport de tests | 9 |
| 4. SONAR..... | 10 |
| 4.1. Installer SonarQube (docker compose)..... | 10 |
| 4.2. Lancer un SonarScanner sur le projet via Jenkins en créant un nouveau job. | 11 |
| 4.3. Parcourir le tableau de bord | 15 |
| 4.4. Modifier le projet de façon à faire varier les indicateurs et relancer une analyse : | 16 |
| 4.4.1. Tests..... | 16 |
| 4.4.2. Duplication..... | 16 |
| 4.4.3. Violations..... | 16 |

1. GIT

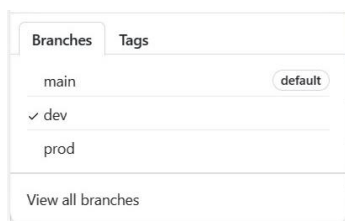
1.1. Créer un repo GIT Ajouter les branches "prod" et "dev" Charger sur la branche "dev" les fichiers du projet démo

J'ai d'abord créé un repository git sur l'interface web de GitHub :



Ensuite j'ai créé les branches :

- Dev
- Prod



Pour charger les fichiers du dossier démo dans la branche « dev » :

dev 3 Branches 0 Tags Add file Code

This branch is 15 commits ahead of main.

Amarakonte ok

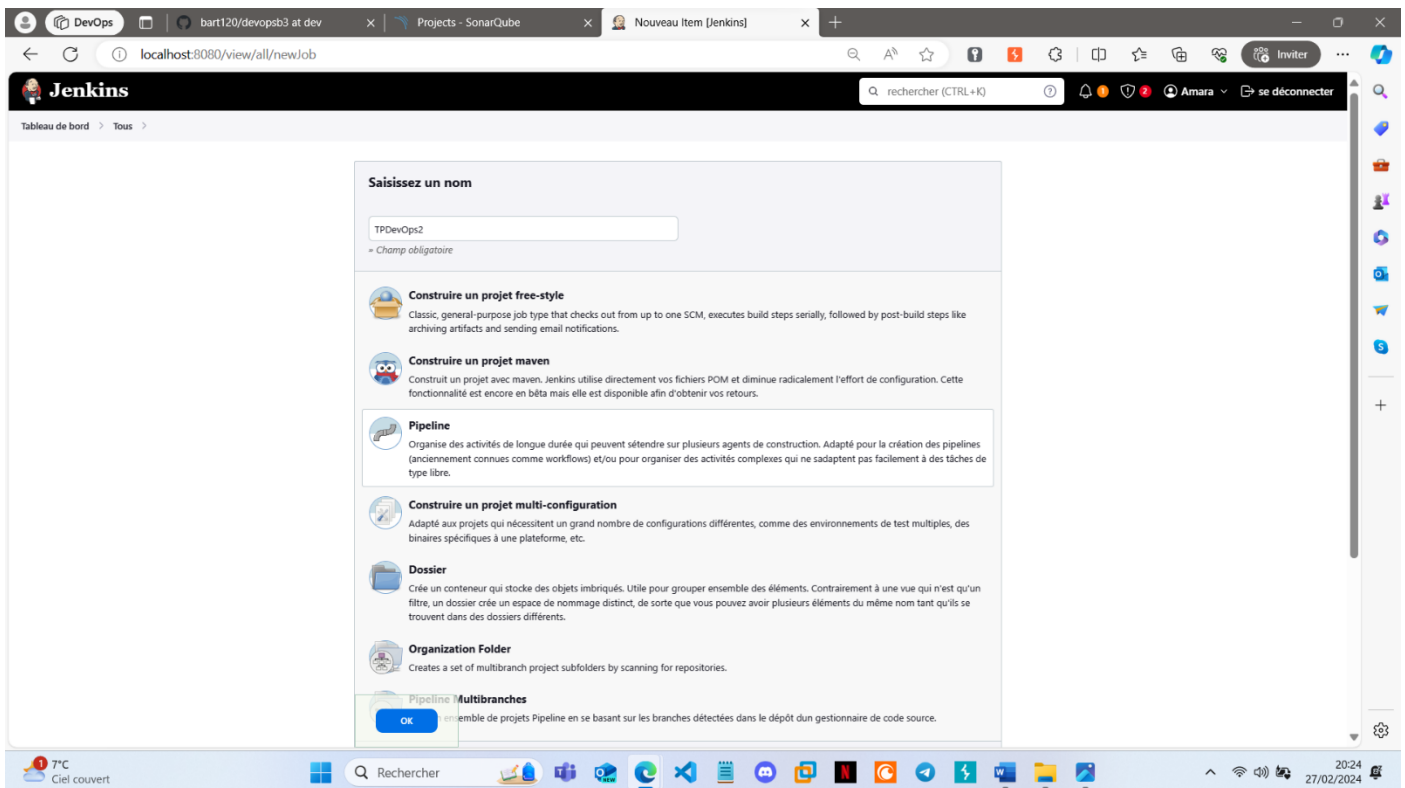
d4ffd05 · 7 hours ago

24 Commits

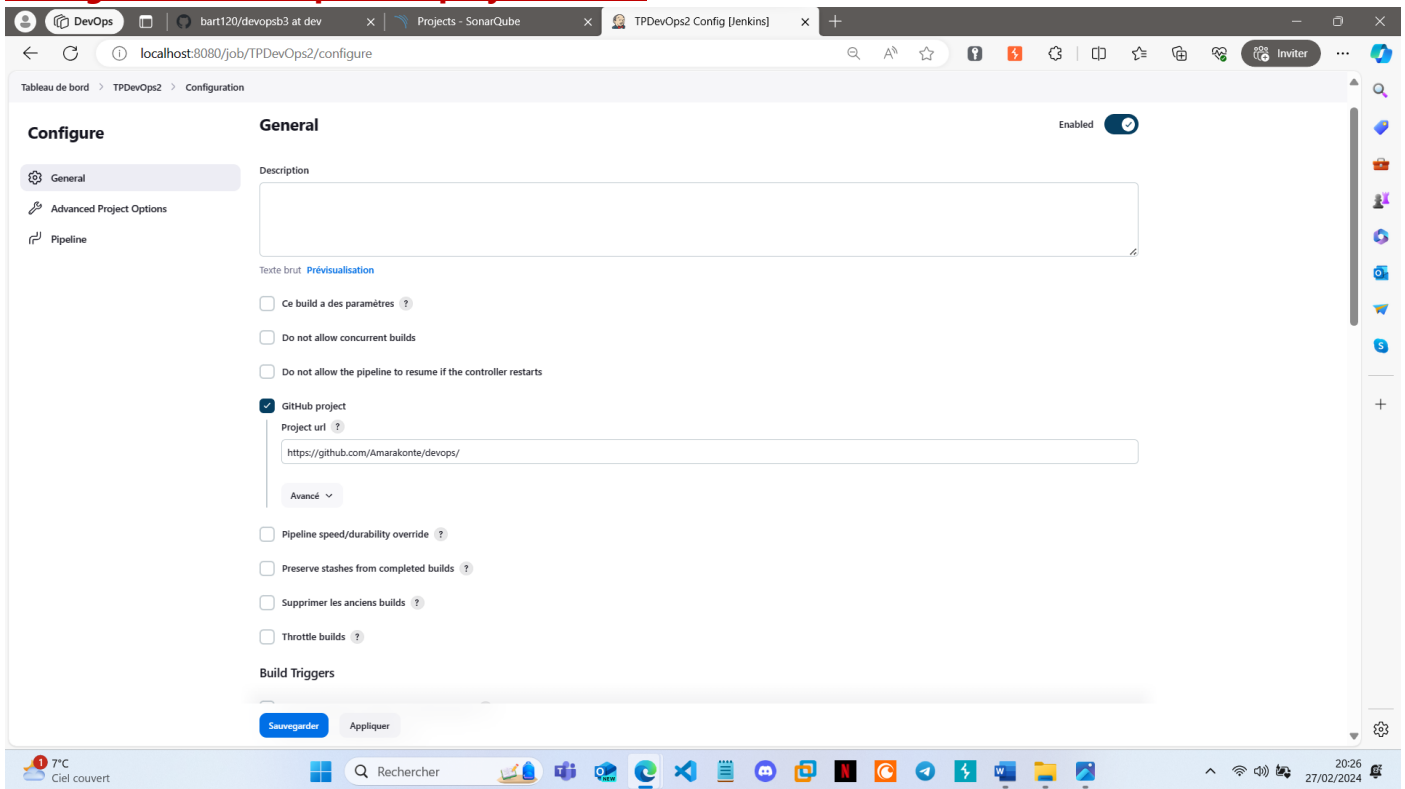
| | | |
|--------------|----------------------------|--------------|
| .mvn/wrapper | v1 | 2 weeks ago |
| src | ok | 7 hours ago |
| .gitignore | v1 | 2 weeks ago |
| Jenkinsfile | ok | 7 hours ago |
| mvnw | v1 | 2 weeks ago |
| mvnw.cmd | v1 | 2 weeks ago |
| pom.xml | test reussi + calculatrice | 20 hours ago |

2. Compilation

2.1. Créer un job Jenkins qui récupère le projet sur le repo GIT et démarre une compilation via Maven.



Configuration : Récupérer le projet GitHub



Configuration : Pipeline Script

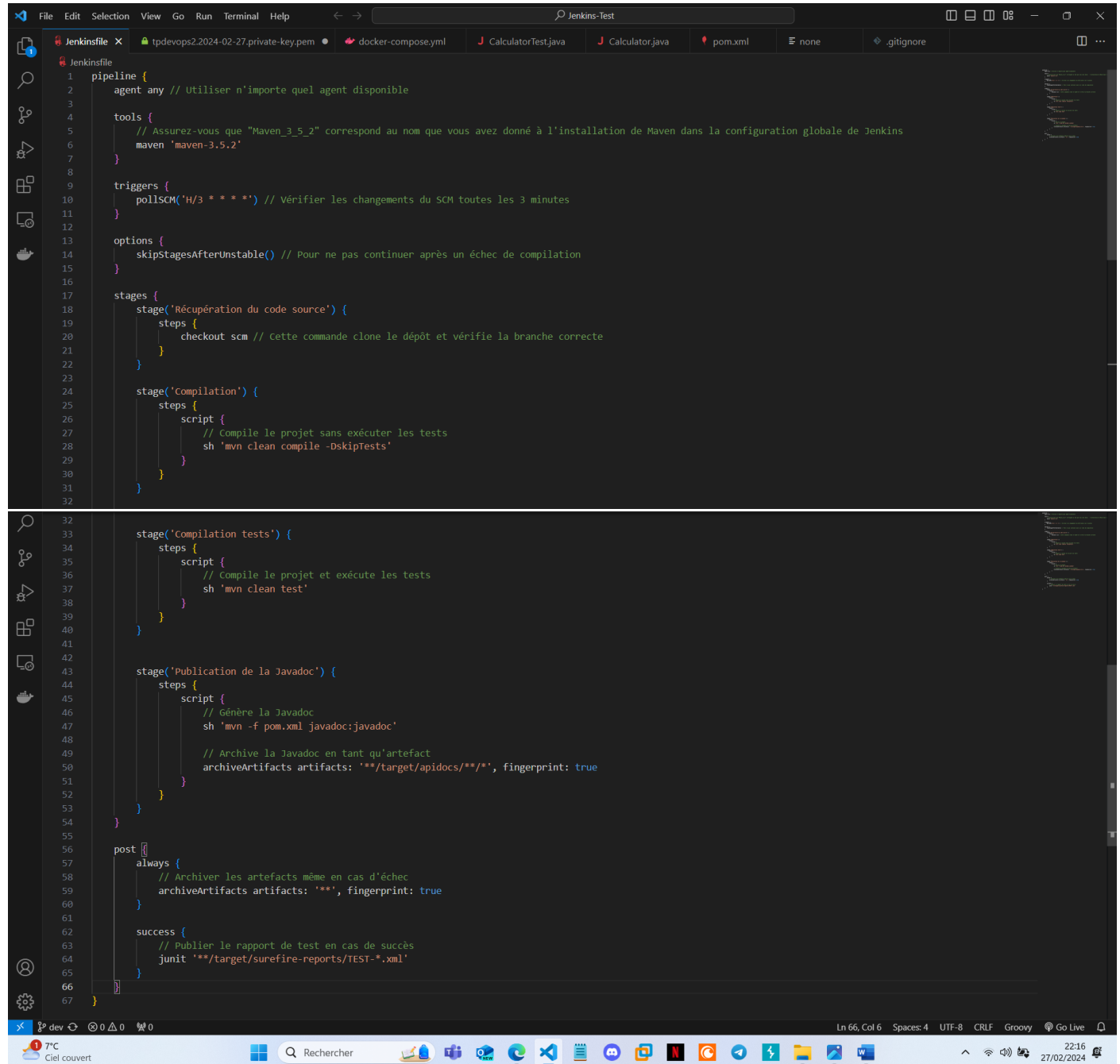
The screenshot shows the Jenkins web interface for configuring a pipeline. The browser address bar indicates the URL is `localhost:8080/job/TPDevOps2/configure`. The page title is "TPDevOps2 Config [Jenkins]".

Configuration Page Structure:

- Left Sidebar:** Contains navigation links: "General", "Advanced Project Options", and "Pipeline" (which is currently selected).
- Main Content Area:**
 - Pipeline Definition:** A dropdown menu set to "Pipeline script from SCM".
 - SCM:** A dropdown menu set to "Git".
 - Repositories:**
 - Repository URL:** A text field containing `https://github.com/Amarakonte/devops.git`.
 - Credentials:** A dropdown menu set to "- aucun -".
 - + Ajouter:** A button to add more repositories.
 - Avancé:** A dropdown menu.
 - Add Repository:** A button to add a new repository.
 - Branches to build:**
 - Branch Specifier (blank for 'any'):** A text field containing `*/dev`.
 - Add Branch:** A button to add a new branch.
 - Navigateur de la base de code:** A dropdown menu set to "(Auto)".
 - Additional Behaviours:** A dropdown menu set to "Ajouter".
 - Script Path:** A text field containing `jenkinsfile`.
 - Lightweight checkout:** A checkbox that is checked.
 - Pipeline Syntax:** A link to view the pipeline syntax.
- Buttons:** At the bottom of the configuration area are two buttons: "Sauvegarder" (Save) and "Appliquer" (Apply).

Footer: The bottom of the page shows the system tray with the date and time "20:27 27/02/2024" and the Jenkins version "Jenkins 2.441".

Jenkinsfile :



```
1 pipeline {
2   agent any // Utiliser n'importe quel agent disponible
3
4   tools {
5     // Assurez-vous que "Maven_3.5.2" correspond au nom que vous avez donné à l'installation de Maven dans la configuration globale de Jenkins
6     maven 'maven-3.5.2'
7   }
8
9   triggers {
10    pollSCM('H/3 * * * *') // Vérifier les changements du SCM toutes les 3 minutes
11  }
12
13  options {
14    skipStagesAfterUnstable() // Pour ne pas continuer après un échec de compilation
15  }
16
17  stages {
18    stage('Récupération du code source') {
19      steps {
20        checkout scm // Cette commande clone le dépôt et vérifie la branche correcte
21      }
22    }
23
24    stage('compilation') {
25      steps {
26        script {
27          // Compile le projet sans exécuter les tests
28          sh 'mvn clean compile -DskipTests'
29        }
30      }
31    }
32
33    stage('Compilation tests') {
34      steps {
35        script {
36          // Compile le projet et exécute les tests
37          sh 'mvn clean test'
38        }
39      }
40    }
41
42    stage('Publication de la Javadoc') {
43      steps {
44        script {
45          // Génère la Javadoc
46          sh 'mvn -f pom.xml javadoc:javadoc'
47
48          // Archive la Javadoc en tant qu'artefact
49          archiveArtifacts artifacts: '**/target/apidocs/**/*', fingerprint: true
50        }
51      }
52    }
53  }
54
55  post {
56    always {
57      // Archiver les artefacts même en cas d'échec
58      archiveArtifacts artifacts: '**/*', fingerprint: true
59    }
60
61    success {
62      // Publier le rapport de test en cas de succès
63      junit '**/target/surefire-reports/TEST-*.xml'
64    }
65  }
66 }
67
```

2.2. Vérifier que le projet compile

Tableau de bord > TPDevOps2

Status TPDevOps2

Changes

Lancer un build

Configurer

Supprimer Pipeline

Full Stage View

GitHub

Renommer

Pipeline Syntax

Log du dernier accès à Git

Historique des builds tendance

Atom feed des builds Atom feed des échecs

Ajouter une description

Désactiver le projet

Last Successful Artifacts

Stage View

| | Declarative: Checkout SCM | Declarative: Tool Install | Récupération du code source | Compilation | Compilation tests | Publication de la Javadoc | Declarative: Post Actions |
|---|---------------------------|---------------------------|-----------------------------|-------------|-------------------|---------------------------|---------------------------|
| Average stage times: (Average full run time ~2min 45s) | 3s | 1s | 4s | 29s | 59s | 50s | 8s |
| #22 Feb. 27 17:19 No Changes | 2s | 1s | 3s | 22s | 36s | 29s | 5s |
| #21 Feb. 27 12:46 1 commit | 2s | 1s | 3s | 26s | 1min 9s | 1min 0s | 10s |
| #20 Feb. 27 12:37 1 commit | 5s | 2s | 7s | 39s | 1min 11s | 1min 1s | 10s |

Liens permanents

- Dernier build (#22), il y a 3 h 11 mn
- Dernier build stable (#22), il y a 3 h 11 mn
- Dernier build avec succès (#22), il y a 3 h 11 mn
- Dernier build en échec (#16), il y a 21 h
- Dernier build non réussi (#16), il y a 21 h
- Dernier build complété (#22), il y a 3 h 11 mn

2.3. Ne lance pas de tests

```
stage('Compilation') {
    steps {
        script {
            // Compile le projet sans exécuter les tests
            sh 'mvn clean compile -DskipTests'
        }
    }
}
```

2.4. Publie la Javadoc

```
stage('Publication de la Javadoc') {
    steps {
        script {
            // Génère la Javadoc
            sh 'mvn -f pom.xml javadoc:javadoc'

            // Archive la Javadoc en tant qu'artefact
            archiveArtifacts artifacts: '**/target/apidocs/**/*', fingerprint: true
        }
    }
}
```

2.5. Se lance toute les 3 min si changement du SCM

```
triggers {  
|   pollSCM('H/3 * * * *') // Vérifier les changements du SCM toutes les 3 minutes  
}
```

3. Tests Maven

3.1. Ajouter JUnit pour faire des tests dans le projet pour créer plusieurs tests et quelques méthodes simples de calcul.

Fichier POM.XML

```
<dependency>  
|   <groupId>junit</groupId>  
|   <artifactId>junit</artifactId>  
|   <version>4.13.2</version>  
|   <scope>test</scope>  
</dependency>
```

Import Java

```
import org.junit.Test;  
import static org.junit.Assert.*;
```

3.2. Integer add(Integer a, Integer b)

```
@Test  
public void testAdd() {  
|   Calculator calculator = new Calculator();  
|   int result = calculator.add(3, 4);  
|   assertEquals(7, result);  
}
```

3.3. Integer multi(Integer a, Integer b)

```
@Test  
public void testMultiply() {  
|   Calculator calculator = new Calculator();  
|   int result = calculator.multiply(3, 4);  
|   assertEquals(12, result);  
}
```

3.4. Integer div(Integer a, Integer b)

```
@Test  
public void testDivide() {  
|   Calculator calculator = new Calculator();  
|   int result = calculator.divide(8, 2);  
|   assertEquals(4, result);  
}
```


3.5. Dans un nouveau job Jenkins :

3.5.1. Vérifier que la compilation et les tests sont passants.

| | Declarative: Checkout SCM | Declarative: Tool Install | Récupération du code source | Compilation | Compilation et tests | Publication de la Javadoc | Declarative: Post Actions |
|---|---------------------------|---------------------------|-----------------------------|-------------|----------------------|---------------------------|---------------------------|
| Average stage times: (Average full run time: ~1min 45s) | 2s | 756ms | 2s | 19s | 33s | 17s | 4s |
| 1 | 2s | 826ms | 2s | 19s | 39s | 34s | 6s |

3.5.2. Modifier un test de manière à la faire échouer.

| | Declarative: Checkout SCM | Declarative: Tool Install | Récupération du code source | Compilation | Compilation et tests | Publication de la Javadoc | Declarative: Post Actions |
|---|---------------------------|---------------------------|-----------------------------|-------------|----------------------|---------------------------|---------------------------|
| Average stage times: (Average full run time: ~1min 45s) | 2s | 756ms | 2s | 19s | 33s | 17s | 4s |
| 1 | 2s | 826ms | 2s | 19s | 39s | 34s | 6s |
| 16 | 2s | 686ms | 2s | 19s | 27s | 567ms | 2s |

DevOps

bart120/devopsb3 at dev

Projects - SonarQube

TPDevOps2 #16 Console [Jenkins: x]

localhost:8080/job/TPDevOps2/16/console

Tableau de bord > TPDevOps2 > #16

```
location: class CalculatorTest
[INFO] 8 errors
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 20.853 s
[INFO] Finished at: 2024-02-26T22:14:44Z
[INFO] Final Memory: 38M/108M
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.11.0:testCompile (default-testCompile) on project demo: Compilation failure: Compilation failure:
[ERROR] /var/jenkins_home/workspace/TPDevOps2/src/test/java/CalculatorTest.java:[1,17] package org.junit does not exist
[ERROR] /var/jenkins_home/workspace/TPDevOps2/src/test/java/CalculatorTest.java:[2,24] package org.junit does not exist
[ERROR] /var/jenkins_home/workspace/TPDevOps2/src/test/java/CalculatorTest.java:[6,6] cannot find symbol
[ERROR] symbol: class Test
[ERROR] location: class CalculatorTest
[ERROR] /var/jenkins_home/workspace/TPDevOps2/src/test/java/CalculatorTest.java:[13,6] cannot find symbol
[ERROR] symbol: class Test
[ERROR] location: class CalculatorTest
[ERROR] /var/jenkins_home/workspace/TPDevOps2/src/test/java/CalculatorTest.java:[20,6] cannot find symbol
[ERROR] symbol: class Test
[ERROR] location: class CalculatorTest
[ERROR] /var/jenkins_home/workspace/TPDevOps2/src/test/java/CalculatorTest.java:[10,9] cannot find symbol
[ERROR] symbol: method assertEquals(int,int)
[ERROR] location: class CalculatorTest
[ERROR] /var/jenkins_home/workspace/TPDevOps2/src/test/java/CalculatorTest.java:[17,9] cannot find symbol
[ERROR] symbol: method assertEquals(int,int)
[ERROR] location: class CalculatorTest
[ERROR] /var/jenkins_home/workspace/TPDevOps2/src/test/java/CalculatorTest.java:[24,9] cannot find symbol
[ERROR] symbol: method assertEquals(int,int)
[ERROR] location: class CalculatorTest
[ERROR] -> [help 1]
[ERROR] -----
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR] -----
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [help 1] http://wiki.apache.org/confluence/display/MAVEN/MojoFailureException
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
```

3.5.3. Vérifier que la compilation fonctionne et que le test remonte en KO.

Stage View



3.5.4. Ignorer le test via Maven de façon à obtenir un build correct.

```
stage('Compilation') {  
    steps {  
        script {  
            // Compile le projet sans exécuter les tests  
            sh 'mvn clean compile -DskipTests'  
        }  
    }  
}
```

3.5.5. Publier le rapport de tests

```
post {  
    always {  
        // Archiver les artefacts même en cas d'échec  
        archiveArtifacts artifacts: '**', fingerprint: true  
    }  
    success {  
        // Publier le rapport de test en cas de succès  
        junit '**/target/surefire-reports/TEST-*.xml'  
    }  
}
```

Artefacts de TPDevOps2 #27

| | | |
|--|-------------------------|-----------|
| / target / surefire-reports / | | → |
| com.example.demo.DemoApplicationTests.txt | 27 févr. 2024, 21:10:41 | 324 B |
| TEST-com.example.demo.DemoApplicationTests.xml | 27 févr. 2024, 21:10:41 | 15.56 KiB |

(Tous les fichiers dans un zip)

4. SONAR

4.1. Installer SonarQube (docker compose).

```
D: > devops > docker-compose.yml
1  version: '3.9'
2
3  networks:
4    devops:
5      name: network-devops
6
7  services:
8    jenkins:
9      container_name: jenkins-cont
10     image: jenkins/jenkins:latest
11     restart: always
12     networks:
13       - devops
14     volumes:
15       - D:\devops\jenkins:/var/jenkins_home
16     ports:
17       - 8080:8080
18       - 50000:50000
19
20     sonar:
21       container_name: sonar-cont
22       image: sonarqube:latest
23       restart: always
24       networks:
25         - devops
26       volumes:
27         - D:\devops\sonar:/opt/sonarqube/data
28       ports:
29         - 9000:9000
30         - 9092:9092
31
32 #   extra_hosts: # pour linux
33 #     - "host.docker.internal:host-gateway"
```

Installer le plugin SonarQube Scanner

The screenshot shows the Jenkins web interface at localhost:8080. The 'Plugins' page is active, displaying a list of available plugins. A red box highlights the 'SonarQube Scanner' plugin, version 2.17.2. The plugin is marked as 'Installable' with a checkmark icon. The description states: 'This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.' The last update time is 'Il y a 7 j 23 h'. Below the plugin entry, there is a warning message: 'Avertissement : cette version de plugin n'est peut-être pas sûre. Veuillez vérifier les notices de sécurité suivantes :'. The warning is dated 'Il y a 5 an. 6 mo.'.

Paramétrer SonarQube Servers dans les config globales

localhost:8080/manage/configure

Tableau de bord > Administrer Jenkins > System

☐ Emplacement des outils
☐ Variables d'environnement

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

☐ Environment variables

Installations de SonarQube

Liste des installations de SonarQube

Nom
SonarQube

URL du serveur
Par défaut à <http://localhost:9000>
http://sonar9000

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
- aucun -

+ Ajouter +

Avancé ▾

Ajouter une installation SonarQube

4.2. Lancer un SonarScanner sur le projet via Jenkins en créant un nouveau job.

Création d'un projet local sur SonarQube

1 of 2

Create a local project

Project display name *

TPDevOps2

Up to 255 characters. Some scanners might override the value you provide.

Project key *

TPDevOps2-

The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

dev

The name of your project's default branch [Learn More](#)

Cancel Next

Issues Rules Quality Profiles Quality Gates Administration More Q

2 of 2

Set up project for Clean as You Code

The new code definition sets which part of your code will be considered new code. This helps you focus attention on the most recent changes to your project, enabling you to follow the Clean as You Code methodology. [Learn more: Defining New Code](#)

Choose the baseline for new code for this project

☒ Use the global setting

Previous version
Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

☐ Define a specific setting for this project

☐ Previous version
Any code that has changed since the previous version is considered new code.
Recommended for projects following regular versions or releases.

☐ Number of days
Any code that has changed in the last x days is considered new code. If no action is taken on a new issue after x days, this issue will become part of the overall code.
Recommended for projects following continuous delivery.

☐ Reference branch
Choose a branch as the baseline for the new code.
Recommended for projects using feature branches.

[Back](#) [Create project](#)

Générer la clé du projet

Analysis Method > Locally

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

[Generate a project token](#) [Use existing token](#)

Token name ?

Analyze "TPDevOps2"

Expires in

30 days

[Generate](#)

Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your [user account](#). See the [documentation](#) for more information.

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

2 Run analysis on your project

Analysis Method > Locally

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Analyze "TPDevOps2": `sqp_1d5b24c60840add0ab042a6809b6238725dd05c0`

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

[Continue](#)

2 Run analysis on your project

2 Run analysis on your project

What option best describes your build?

[Maven](#) [Gradle](#) [.NET](#) [Other \(for JS, TS, Go, Python, PHP, ...\)](#)

Execute the Scanner for Maven

Running a SonarQube analysis with Maven is straightforward. You just need to run the following command in your project's folder.

```
mvn clean verify -Dsonar.projectKey=TPDevOps2 -Dsonar.projectName="TPDevOps2" -Dsonar.host.url=http://localhost:9000 -Dsonar.token=sqp_1d5b24c60840add0ab042a6809b6238725dd05c0
```

[Copy](#)

Please visit the [official documentation of the Scanner for Maven](#) for more details.

Création job sur Jenkins

The screenshot shows the Jenkins web interface at localhost:8080. The 'Nouveau Item [Jenkins]' tab is active. The 'Saisissez un nom' field contains 'TPDevOps2-Scanner'. Below the field, a list of job types is displayed:

- Construire un projet free-style**: Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.
- Construire un projet maven**: Construit un projet avec maven. Jenkins utilise directement vos fichiers POM et diminue radicalement l'effort de configuration. Cette fonctionnalité est encore en bêta mais elle est disponible afin d'obtenir vos retours.
- Pipeline**: Organise des activités de longue durée qui peuvent s'étendre sur plusieurs agents de construction. Adapté pour la création des pipelines (anciennement connues comme workflows) et/ou pour organiser des activités complexes qui ne s'adaptent pas facilement à des tâches de type libre.
- Construire un projet multi-configuration**: Adapté aux projets qui nécessitent un grand nombre de configurations différentes, comme des environnements de test multiples, des binaires spécifiques à une plateforme, etc.
- Dossier**: Crée un conteneur qui stocke des objets imbriqués. Utile pour grouper ensemble des éléments. Contrairement à une vue qui n'est qu'un filtre, un dossier crée un espace de nommage distinct, de sorte que vous pouvez avoir plusieurs éléments du même nom tant qu'ils se trouvent dans des dossiers différents.
- Organization Folder**: Creates a set of multibranch project subfolders by scanning for repositories.
- Pipeline Multibranches**: Crée un ensemble de projets Pipeline en se basant sur les branches détectées dans le dépôt d'un gestionnaire de code source.

The 'OK' button is highlighted in blue. The bottom of the page shows a Windows taskbar with the date 27/02/2024 and time 21:22.

Configuration : Pipeline Script

The screenshot shows the Jenkins configuration page for a pipeline script. The page is titled "Configure" and has a sidebar with "General", "Advanced Project Options", and "Pipeline". The "General" tab is selected, and the "Pipeline" section is expanded. The "Definition" dropdown is set to "Pipeline script". The "Script" field contains a Groovy script for a pipeline. The script defines a pipeline with a single stage named "Build". The stage has a step named "Build" that runs a Maven command to install the Maven version configured as "M3" and add it to the path. The script also includes a "post" block that runs a command to run Maven on a Windows agent, using the "bat" command. The "Use Groovy Sandbox" checkbox is checked. The "Apply" button is visible at the bottom right.

Configure

General Enabled

Description

Texte brut [Prévisualisation](#)

☐ Ce build a des paramètres ?

☐ Do not allow concurrent builds

☐ Do not allow the pipeline to resume if the controller restarts

☒ **GitHub project**

Project url ?

<https://github.com/Amarakonte/devops/>

Avancé ▾

☐ Pipeline speed/durability override ?

☐ Preserve stashes from completed builds ?

☐ Supprimer les anciens builds ?

☐ Throttle builds ?

Pipeline

Definition

Pipeline script ▾

Script ?

```
1 pipeline {
2   agent any
3
4   tools {
5     // Install the Maven version configured as "M3" and add it to the path.
6     maven "maven-3.5.2"
7   }
8
9   stages {
10    stage('Build') {
11      steps {
12        // Get some code from a Github repository
13        git branch: 'dev', url: 'https://github.com/Amarakonte/devops.git'
14
15        // Run Maven on a unix agent.
16        sh '''mvn clean verify sonar:sonar \
17          -Dsonar.projectKey=TPDevOps2 \
18          -Dsonar.projectName=TPDevOps2 \
19          -Dsonar.host.url=http://sonar:9000 \
20          -Dsonar.token=5q_10524c6848a0d08042688906238725d095c0'''
21
22        // To run Maven on a Windows agent, use
23        // bat "mvn -Dmaven.test.failure.ignore=true clean package"
24      }
25    }
26
27    post {
28      // If Maven was able to run the tests, even if some of the test
29      // failed, record the test results and archive the jar file.
30      success {
31        junit '**/target/surefire-reports/TEST-*.xml'
32        archiveArtifacts 'target/*.jar'
33      }
34    }
35  }
36 }
37 }
```

☒ Use Groovy Sandbox ?

[Sauvegarder](#) [Appliquer](#)

Lancement des build : réussi

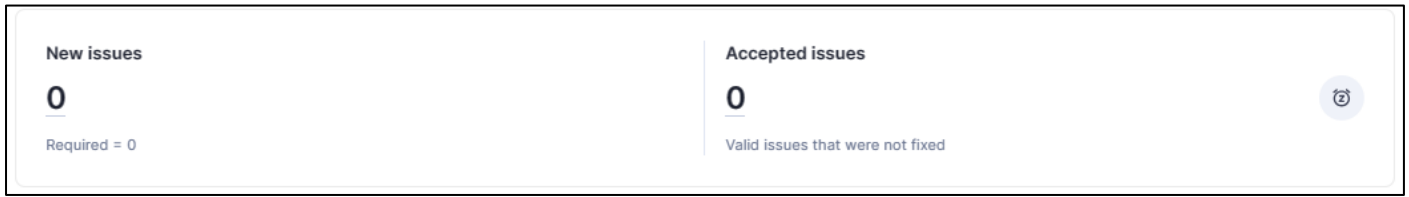
| | | | Declarative: Tool Install | Build |
|---|-------------------|------------|------------------------------|----------|
| Average stage times: (Average <u>full</u> run time: ~1min 20s) | | | 1s | 46s |
| #38 | févr. 28 01:16 | 7 commits | 2s | 2min 38s |
| #37 | févr. 27 23:31 | No Changes | 786ms | 47s |
| #36 | févr. 27 23:27 | No Changes | 862ms | 56s |

4.3. Parcourir le tableau de bord

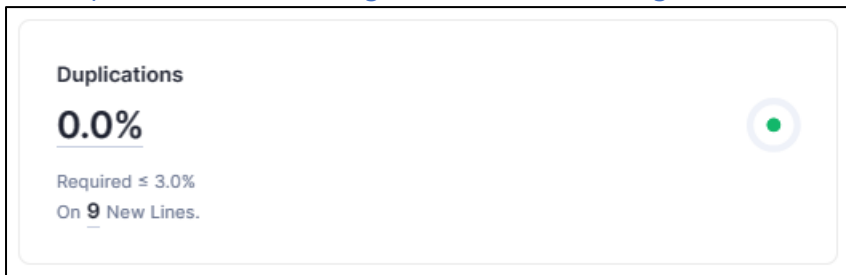
The screenshot displays the SonarQube dashboard for the project 'TPDevOps2' in the 'dev' branch. The main status is 'Passed', indicating that the code quality checks have been successful. The dashboard provides a comprehensive overview of the project's health, including metrics on code quality, security, and reliability. Key data points include 37 Lines of Code, 0 Open Issues, 0 Accepted Issues, 0.0% Duplications, and 0 Security Hotspots. The 'Measures' section further breaks down these metrics into categories like Security, Reliability, and Maintainability. The interface is user-friendly, with clear navigation tabs and a clean layout. The bottom of the image shows a Windows taskbar with various application icons and system information, including the date and time.

4.4. Modifier le projet de façon à faire varier les indicateurs et relancer une analyse :**4.4.1. Tests**

Je n'ai pas réussi à faire bouger cet indicateur, malgré les tests.

**4.4.2. Duplication**

Je n'ai pas réussi à faire bouger cet indicateur, malgré les tests.

**4.4.3. Violations**

Je n'ai pas réussi à faire bouger cet indicateur, malgré les tests.

