



Les tests logiciels

2025

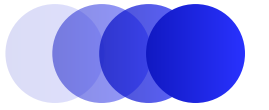
Réalisé par :

Sylla Nfaly

Azzouz Chaimae

Encadré par :

M. Ghailani



01

Introduction

02

Responsabilités de R. QA

03

Les types de tests

04

Outils de test, qualité et
CI/CD

05

Simulation

06

Q. et R. / Conclusion

Missions des responsables QA

1. Définir la stratégie de test

- Sélectionner les types de tests à réaliser (fonctionnels, de performance, de sécurité...).
- Adapter la stratégie au planning, aux ressources et aux exigences du produit.
- Déterminer les priorités de test selon les risques.



Missions des responsables QA

2. Préparer les plans de test

- Rédiger des scénarios de test clairs et précis.
- Définir les données nécessaires pour les tests.
- Identifier les prérequis et les conditions d'exécution.



Missions des responsables QA

3.Choisir les outils de test

But : Utiliser les bons outils pour garantir l'efficacité des tests.

Comment ?

- Comparer et sélectionner des outils adaptés (Selenium, Postman, etc.).
- Configurer les outils selon les besoins du projet.



Missions des responsables QA

4. Automatiser les tests

But : Gagner du temps et éviter les erreurs manuelles.

Comment ?

- Créer ou superviser les scripts d'automatisation.
- Choisir les cas de test à automatiser.
- Intégrer l'automatisation dans la chaîne CI/CD.



Missions des responsables QA

5. Analyser les résultats de test

But : Vérifier la qualité du produit et détecter les anomalies.

Comment ?

- Lire les rapports de test et identifier les échecs.
- Reproduire et documenter les bugs détectés.
- Partager les résultats avec l'équipe projet.



Les types de tests



Il sert à vérifier chaque “petit morceau” du code, comme une fonction ou une méthode.

En gros, on regarde si chaque brique fonctionne toute seule



Là, on ne teste plus juste le code, mais le comportement global du logiciel.

On vérifie si le système fait ce qu’il est censé faire, du point de vue de l’utilisateur.

What is Integration Testing?

Une fois que les “petites briques” ont été testées individuellement, on les met ensemble.

Le test d’intégration vérifie si les différentes parties du système communiquent bien entre elles.

Les types de tests



Test de charge

Ici, on teste comment le système réagit quand il y a beaucoup d'utilisateurs ou beaucoup de données. Le but, c'est de savoir si le logiciel tient le coup, ou s'il "crashe" sous la pression.



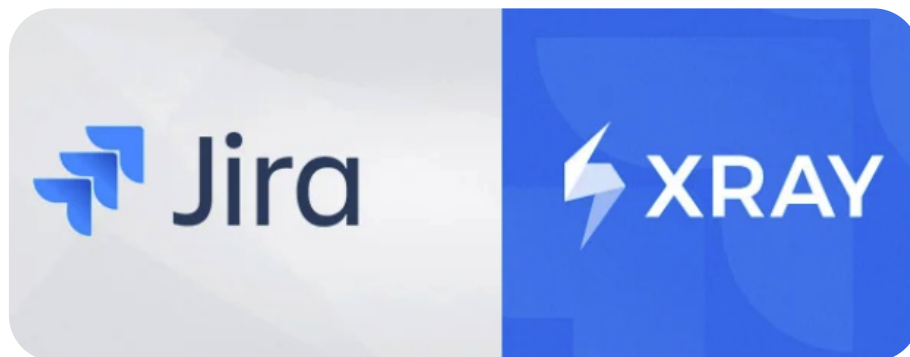
Test de sécurité

Ce test vise à vérifier que le système est protégé contre les attaques ou les accès non autorisés. C'est très important, surtout dans les applis qui traitent des données sensibles.

Outils de test, qualité et CI/CD



Découvrir les outils clés pour assurer la qualité logicielle et automatiser les tests de manière fluide.



Outils de test, qualité et CI/CD

JIRA - XRAY

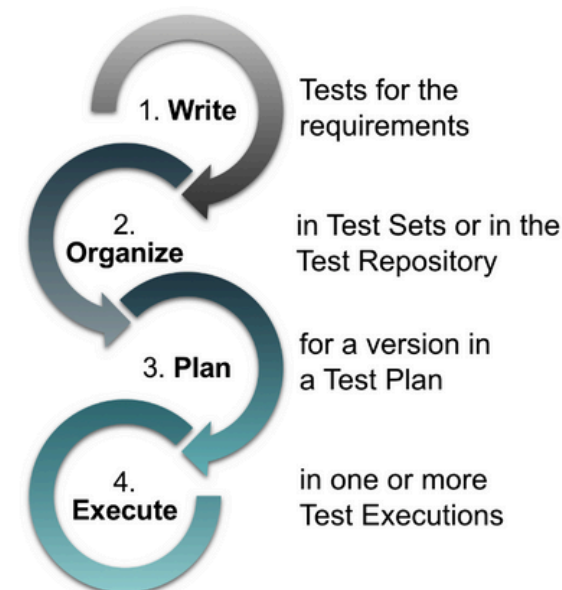
Jira Software est un outil de gestion de projet développé par Atlassian, très utilisé par les équipes de développement logiciel. Il permet de suivre efficacement l'avancement des projets grâce à des tableaux de bord personnalisés et une gestion agile (Kanban, Scrum). Il aide à :

- Organiser les équipes et faciliter la communication
- Accélérer la livraison des projets
- Améliorer les processus en continu
- Simplifier la collaboration entre membres de l'équipe

Xray est un outil complet de gestion de tests intégré à Jira, qui fonctionne de manière autonome sans besoin d'autres logiciels. Il couvre tout le cycle de vie des tests :

- Planification
- Conception
- Exécution
- Reporting

Xray utilise des types de tickets Jira spécifiques pour gérer chacune de ces étapes.



Outils de test, qualité et CI/CD

JIRA - XRAY



◆ Création de Tests

- Les tests sont des tickets Jira de type spécial "Test".
- Types de tests pris en charge :
 - Manuels : avec des étapes détaillées.
 - Automatisés : via Github action ou import de résultats depuis des frameworks externes.
- Possibilité de définir des Préconditions, partagées entre plusieurs tests.

◆ Organisation des Tests

- Suite de Tests : groupe de tests, sous forme de liste, réutilisable.
- Un test peut appartenir à plusieurs suites.
- Bibliothèque de Tests : vue hiérarchique (dossiers/ sous-dossiers) pour mieux organiser.

◆ Planification des Tests

- S'effectue via des Plans de Test.
- Permet de définir :
 - Objectifs de validation.
 - Priorités.
 - Ressources et responsables.
- Le tableau du Plan de Test affiche l'état d'exécution et l'organisation hiérarchique des tests.

◆ Exécution des Tests

- L'Exécution des Tests est une tâche Jira spécifique qui regroupe des tests (manuels ou automatisés).
- Elle collecte les résultats selon :
 - La version du projet.
 - L'environnement d'exécution.
- Il existe aussi les Sous-Exécutions, liées à une exigence spécifique.

Outils de test, qualité et CI/CD

JIRA - XRAY

	H23						
	A	B	C	D	E	F	G
1	TCID	Summary	Description	Test Type	Repository		
2	100	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
3	101	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
4	102	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
5	103	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
6	104	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
7	105	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
8	106	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
9	107	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
10	108	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
11	109	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
12	110	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
13	111	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		
14	112	Tests unitaire	Vérifier qu'il	Automation	Tests unitaire et integations		

Le fichier excel des tests

WebAppProject
Projet logiciel

Sprints actifs

Rapports

Liste

Tickets

Composants

DÉVELOPPEMENT

Code

Livraisons

Pages de projet

Tableau du Test

Projets / WebAppProject / Bibliothèque de Test

La Bibliothèque de Test de projet WebAppProject

DOSSIERS

SUITES DE TESTS

La Bibliothèque de Test

0 (114)

Tests unitaire et integations

24 (24)

Tests d'utilisabilité

18 (18)

Tests fonctionnels

22 (22)

Test de compatibilité

4 (4)

Test de base de données

23 (23)

Tests de sécurité

14 (14)

Tests de performances

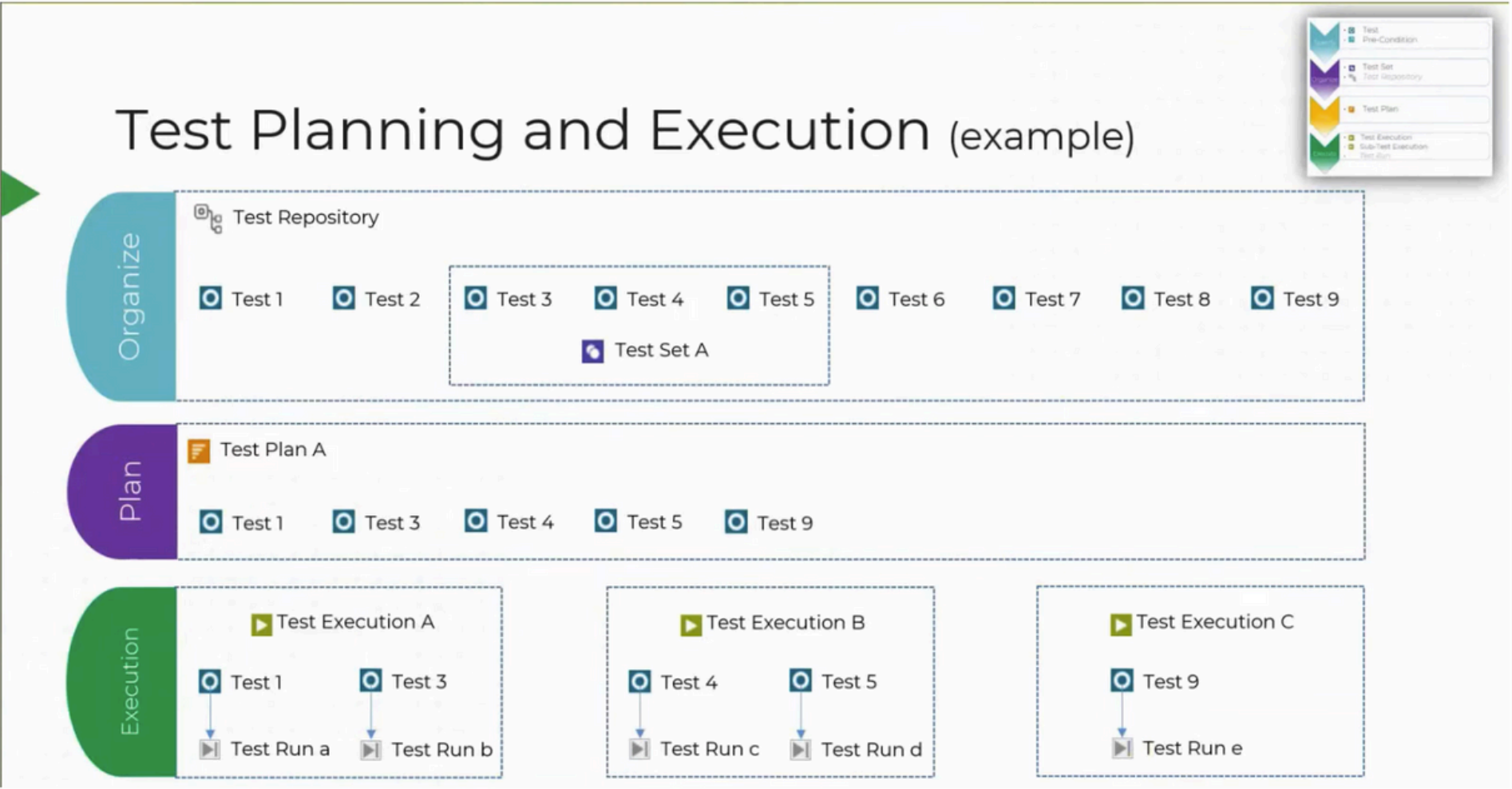
9 (9)

La Bibliothèque d...

Recherche

Afficher 0 de 0 entrées

La bibliothèque des tests.



Planification et d'exécution de tests

Outils de test, qualité et CI/CD

GitHub Actions

Qu'est-ce que GitHub Actions ?

- Service d'automatisation intégré à GitHub
- Permet d'exécuter des workflows (scénarios automatisés)
- Déclencheurs :
 - push, pull_request, schedule, workflow_dispatch
- Cas d'usage :
 - Exécuter les tests (Jest, Cypress)
 - Déployer sur un serveur ou un service (Vercel, Netlify, etc.)
 - Publier un package

Structure d'un workflow

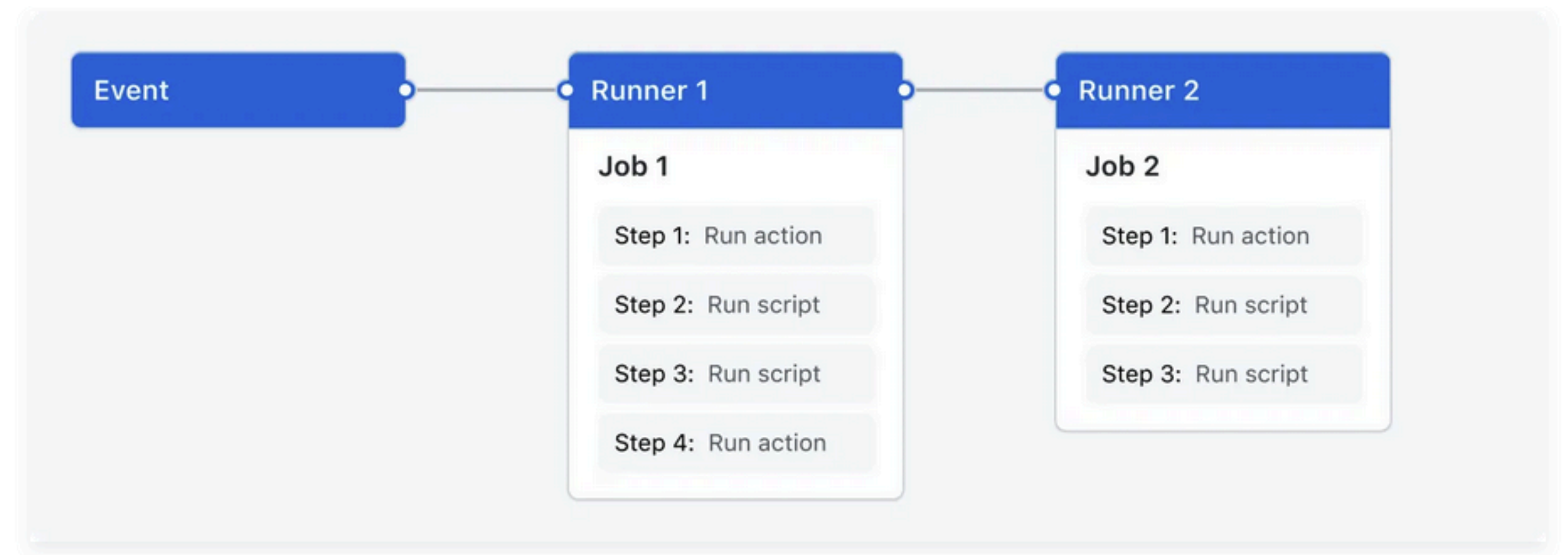
```
1  # .github/workflows/tests.yml
2  name: Run Tests
3
4  on: [push, pull_request]
5
6  jobs:
7    test:
8      runs-on: ubuntu-latest
9
10     steps:
11       - uses: actions/checkout@v3
12       - name: Setup Node.js
13         uses: actions/setup-node@v4
14         with:
15           node-version: '18'
16       - run: npm install
17       - run: npm run test:unit      # Jest
18       - run: npm run test:e2e      # Cypress
19
20
21
```

Outils de test, qualité et CI/CD

GitHub Actions

Structure d'un workflow

```
1  # .github/workflows/tests.yml
2  name: Run Tests
3
4  on: [push, pull_request]
5
6  jobs:
7    test:
8      runs-on: ubuntu-latest
9
10     steps:
11       - uses: actions/checkout@v3
12       - name: Setup Node.js
13         uses: actions/setup-node@v4
14         with:
15           node-version: '18'
16       - run: npm install
17       - run: npm run test:unit      # Jest
18       - run: npm run test:e2e      # Cypress
19
20
21
```



Outils de test, qualité et CI/CD

Jest et @vue/test-utils

Pourquoi Jest ?

- Framework de test JavaScript populaire
- Intégration facile avec Vue.js
- Rapide, simple à configurer
- Prend en charge :
 - Tests synchrones/asynchrones
 - Mocks
 - Snapshots

Installation avec Vue.js

Créer fichier jest.config.js

```
1  module.exports = {  
2    transform: {  
3      '^.+\\.vue$': 'vue-jest',  
4      '^.+\\.js$': 'babel-jest'  
5    }  
6  };  
7  
8
```

```
npm install --save-dev jest @vue/test-utils vue-jest babel-jest
```

Outils de test, qualité et CI/CD

Jest et @vue/test-utils

Les concepts Jest essentiels

- **describe()** : Groupe de tests
- **test()** / **it()** : Un test unique

```
15 >> describe('Exemple', () : void => {
16 >     it('additionne', () : void => {
17         expect(2 + 2).toBe(expected: 4);
18     });
19 });
```

Mock Functions

```
32     const mockFn = jest.fn();
33     mockFn.mockReturnValue(value: 'mocked');
34
35     expect(mockFn()).toBe(expected: 'mocked');
36 );
```

Les Matchers

```
20     expect(value).toBe(expected: 4);
21     expect(value).toEqual(expected: {});
22     expect(value).not.toBe(expected: null);
23     expect(value).toBeNull();
24     expect(value).toBeDefined();
25     expect(value).toBeTruthy();
26     expect(value).toBeFalsy();
27     expect('test string').toMatch(expected: /string/);
28     expect([1,2,3]).toContain(expected: 2);
29 );
```

Outils de test, qualité et CI/CD

Jest et @vue/test-utils



@vue/test-utils

- Bibliothèque officielle pour tester les composants Vue
- **Fonctions clés** : mount, shallowMount, find, trigger, setData, props

```
2
3 import { mount } from '@vue/test-utils';
4
5
```

Test d'intégration avec Jest

- Simule interaction entre parent et enfant
- Vérifie le comportement global

```
43
44 const wrapper : VueWrapper<...> = mount(ParentComponent);
45 await wrapper.find( selector: 'button' ).trigger( eventString: 'click' );
46 expect(wrapper.text()).toContain( expected: 'Formulaire soumis' );
47
48
```

mount vs shallowMount

- mount : monte le composant avec tous ses enfants (test d'intégration)
- shallowMount : remplace les composants enfants par des "stubs" (test unitaire)

```
39
40 mount(App) // tous les composants sont rendus
41 shallowMount(App) // enfants non rendus
42
```

Outils de test, qualité et CI/CD

Tests End-to-End avec Cypress

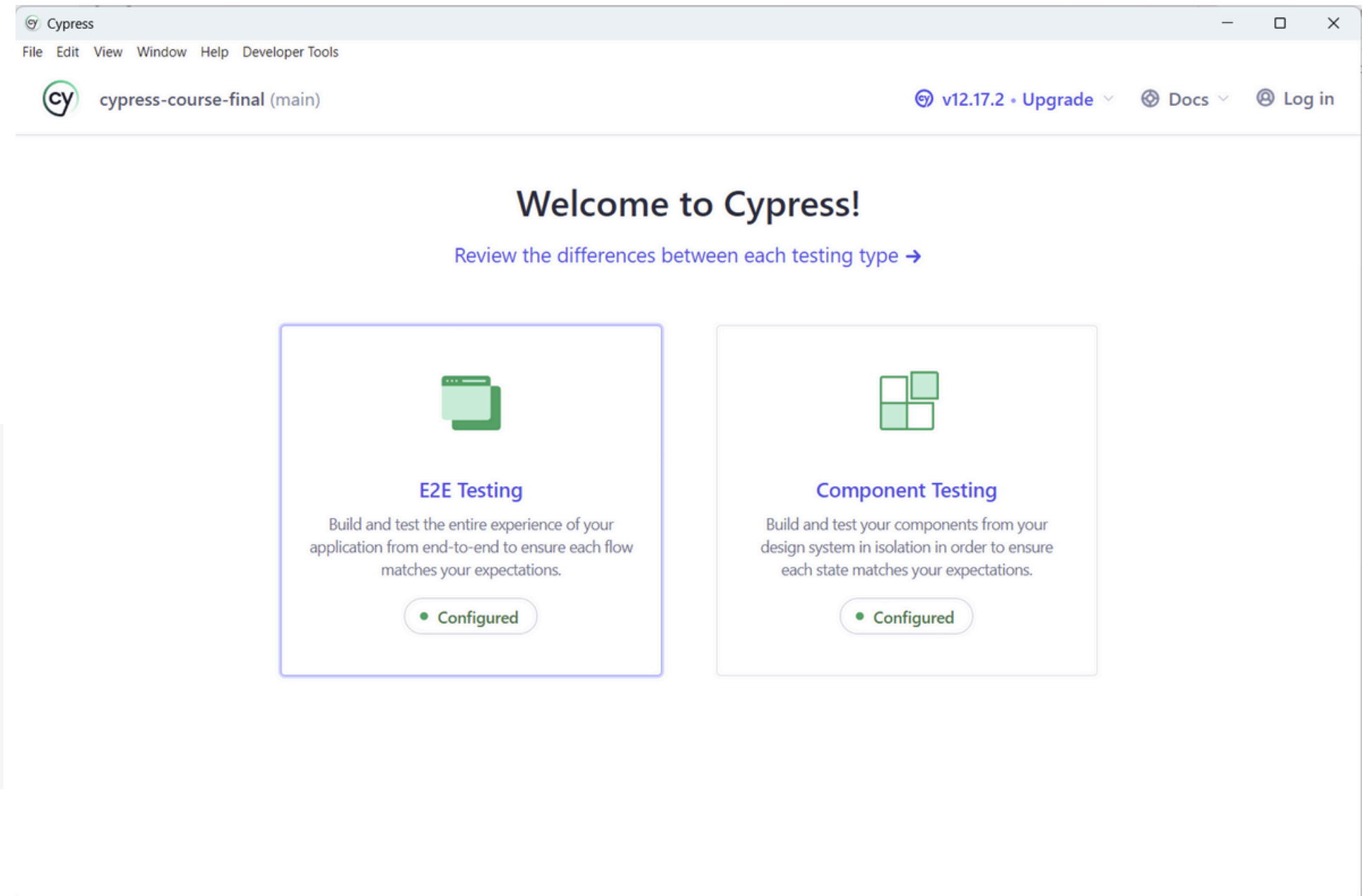
```
npm install cypress --save-dev
```

@vue/test-utils

```
npx cypress open
```

Qu'est-ce que Cypress ?

- Framework de test end-to-end (E2E)
- Exécute les tests dans le navigateur
- Idéal pour tester l'application complète
- Simule un vrai utilisateur : clics, scroll, formulaires, navigation...



Outils de test, qualité et CI/CD

Tests End-to-End avec Cypress



Architecture

Architecture Cypress

Élément	Description
Test Runner	Interface où les tests sont exécutés
Spec Files	Fichiers .cy.js ou .cy.ts avec les scénarios
DOM réel	Tests s'exécutent sur un DOM réel (pas simulé)
Auto-reload	Chaque modification relance les tests

Structure de dossier Cypress

```
cypress/  
├─ e2e/           # Fichiers de tests  
├─ fixtures/      # Données mockées (JSON)  
└─ support/       # Fonctions globales
```

- **cypress.config.js** pour les options globales

Outils de test, qualité et CI/CD

Tests End-to-End avec Cypress



Premier test

```
35
36 >> describe('Ma page d'accueil', fn: () : void => {
37 >   it('doit afficher le titre', config: () : void => {
38     cy.visit('http://localhost:3000');
39     cy.contains('Bienvenue');
40   });
41 });
42
```

- **Cypress** fournit une syntaxe intuitive : `cy.visit`, `cy.get`, `cy.contains`, etc.

Outils de test, qualité et CI/CD

Tests End-to-End avec Cypress



Commandes de base

Commande	Action
cy.visit(url)	Aller sur une page
cy.get(selector)	Sélectionner un élément
cy.contains(text)	Chercher un texte
cy.click()	Cliquer
cy.type('text')	Écrire dans un champ
cy.should()	Assertion

Outils de test, qualité et CI/CD

Tests End-to-End avec Cypress



Assertions

```
38
39 cy.get(selector: 'h1').should(chainer: 'contain', value: 'Bienvenue');
40 cy.get(selector: 'input').should(chainer: 'be.visible');
41 cy.url().should(chainer: 'include', value: '/dashboard');
42
43
```

- **Cypress** intègre Chai pour les assertions (should, expect, etc.)

Outils de test, qualité et CI/CD

Tests End-to-End avec Cypress



Test de formulaire

```
cy.get(selector: 'input[name="email"]').type(text: 'test@email.com');  
cy.get(selector: 'input[name="password"]').type(text: '123456');  
cy.get(selector: 'form').submit();  
cy.contains('Connexion réussie');
```

- Simule un flux complet utilisateur

Outils de test, qualité et CI/CD

Tests End-to-End avec Cypress



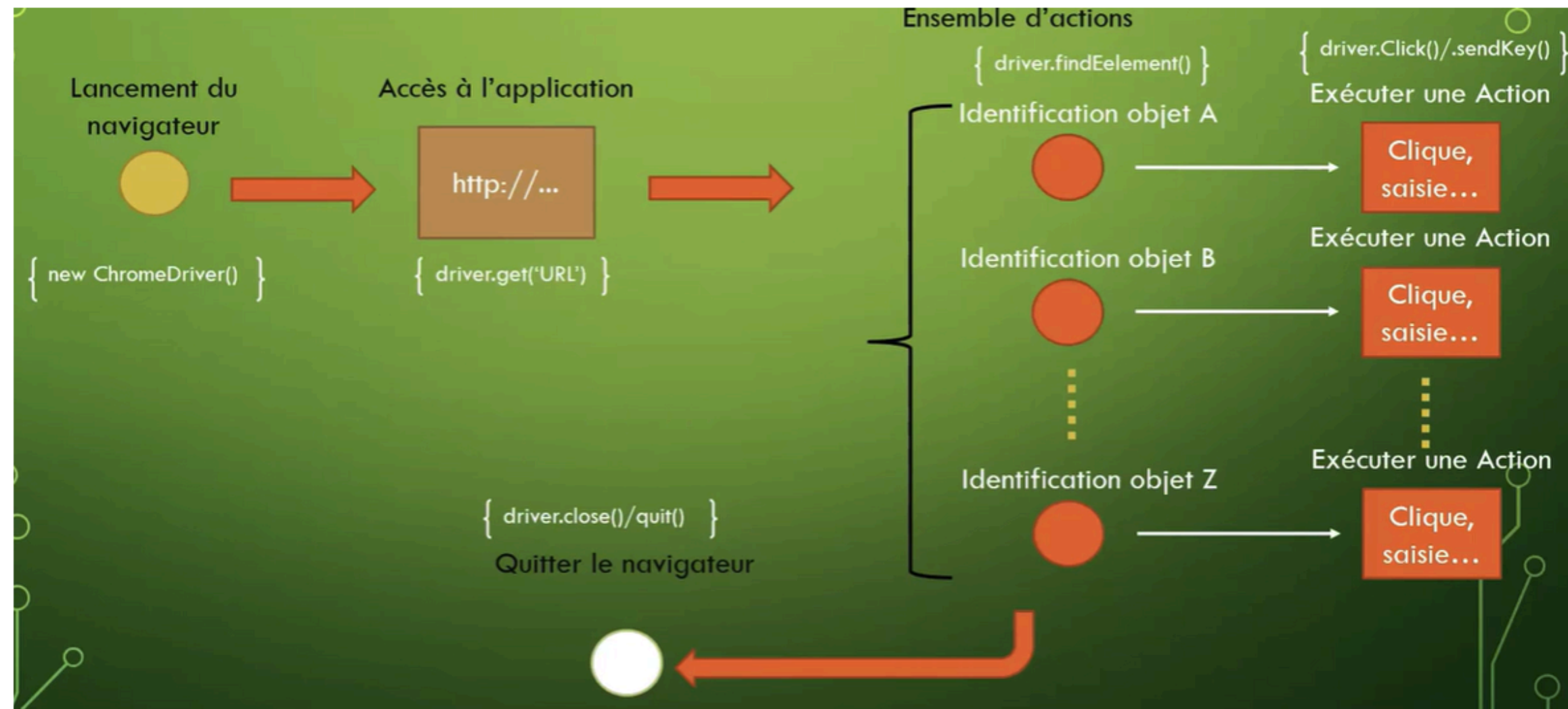
Avant / Après tests (setup / teardown)

```
38
39  beforeEach(fn: () => {
40    cy.visit(url: '/login');
41  });
42
43  afterEach(fn: () => {
44    // Nettoyage ou déconnexion
45  });
46
```

- Idéal pour réinitialiser l'état entre chaque test

Outils de test, qualité et CI/CD

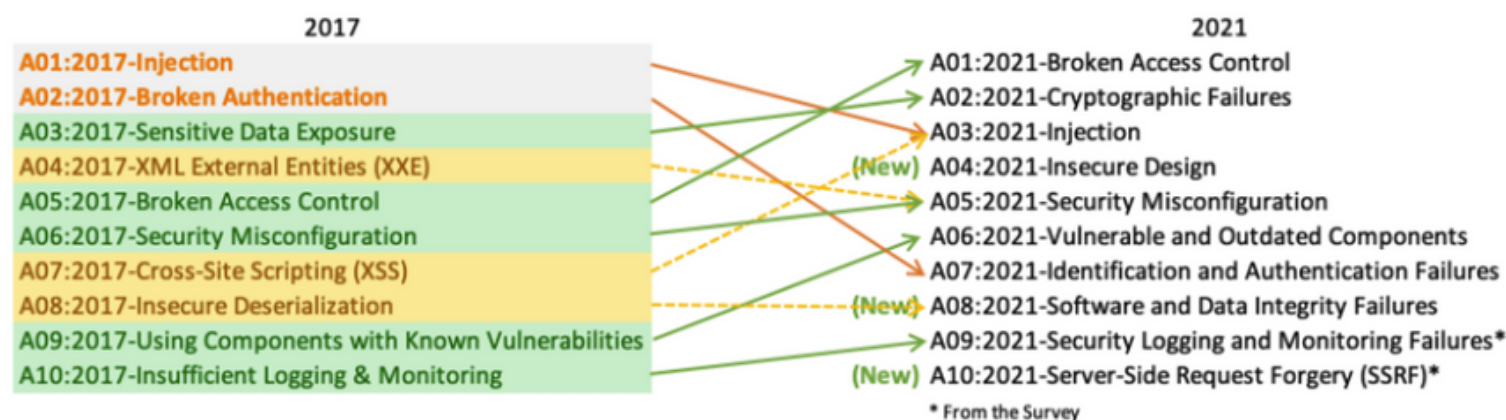
Tests End-to-End avec Cypress



Outils de test, qualité et CI/CD

OWASP ZAP

- 🔒 OWASP – Sécurité des applications web
 - OWASP (Open Web Application Security Project) est une organisation mondiale à but non lucratif, créée en 2001.
 - Elle se consacre à l'amélioration de la sécurité des logiciels.
 - Elle fournit :
 - Des ressources éducatives,
 - Des outils pratiques pour les développeurs et entreprises,
 - Des guides pour atténuer les risques de sécurité.



- ↑
TOP
- Projet phare : OWASP Top 10
- Liste des 10 failles de sécurité web les plus critiques.
 - Mise à jour régulièrement selon l'évolution des menaces.
 - Référence clé pour les développeurs et professionnels de la cybersécurité.

Outils de test, qualité et CI/CD

OWASP ZAP



- 🔒 OWASP ZAP – Audit de sécurité web
 - OWASP ZAP (Zed Attack Proxy) est un outil gratuit d'analyse de sécurité des applications web.
 - Développé par l'organisation OWASP, il aide à identifier les failles de sécurité courantes.

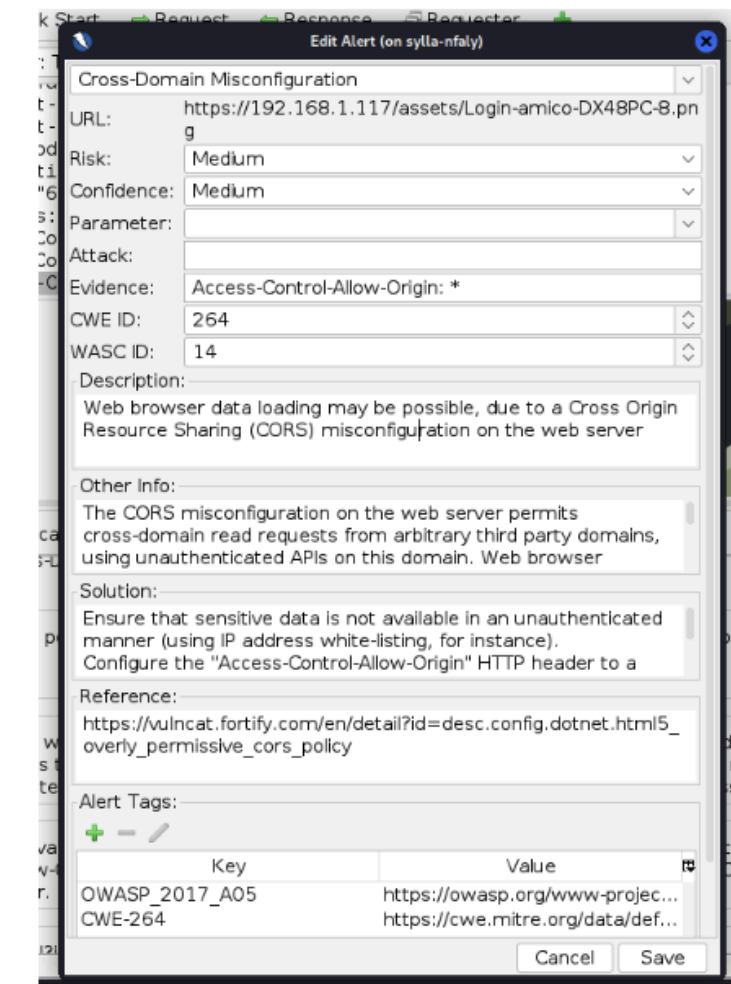
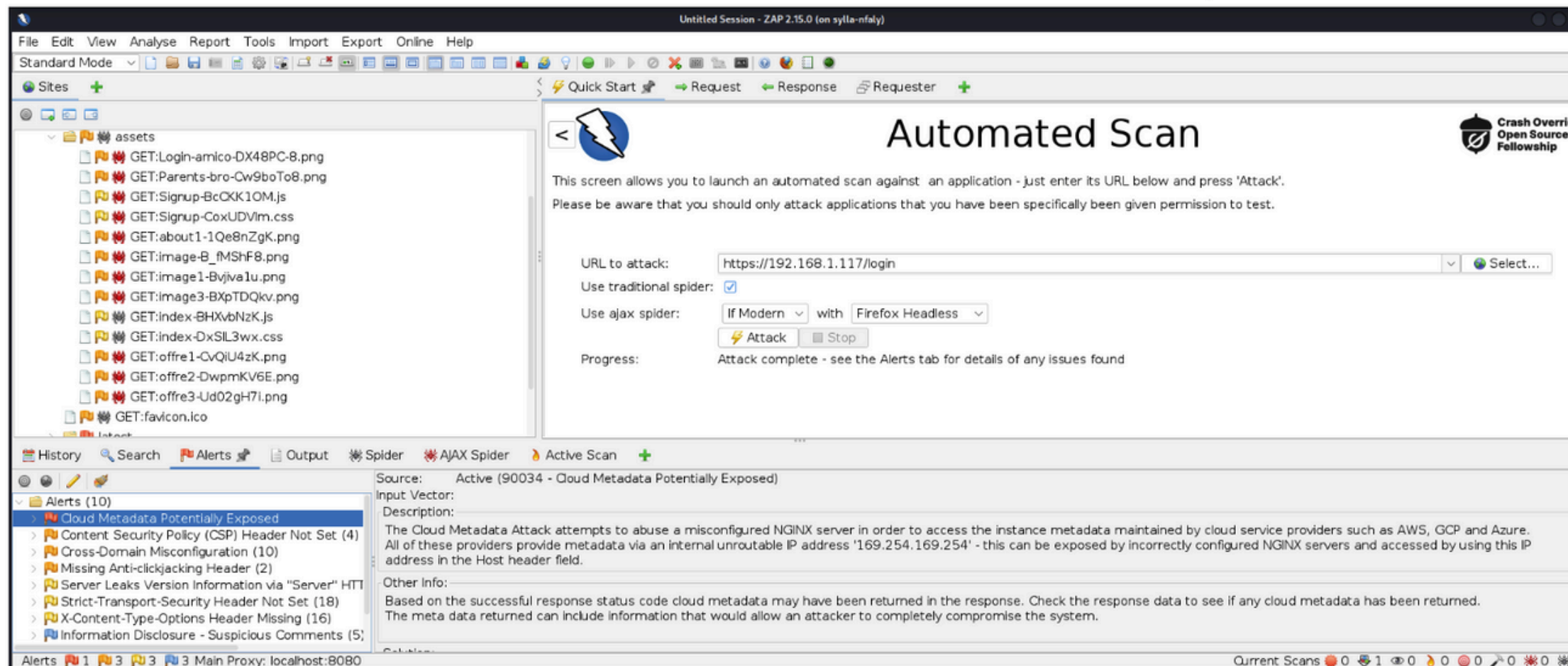
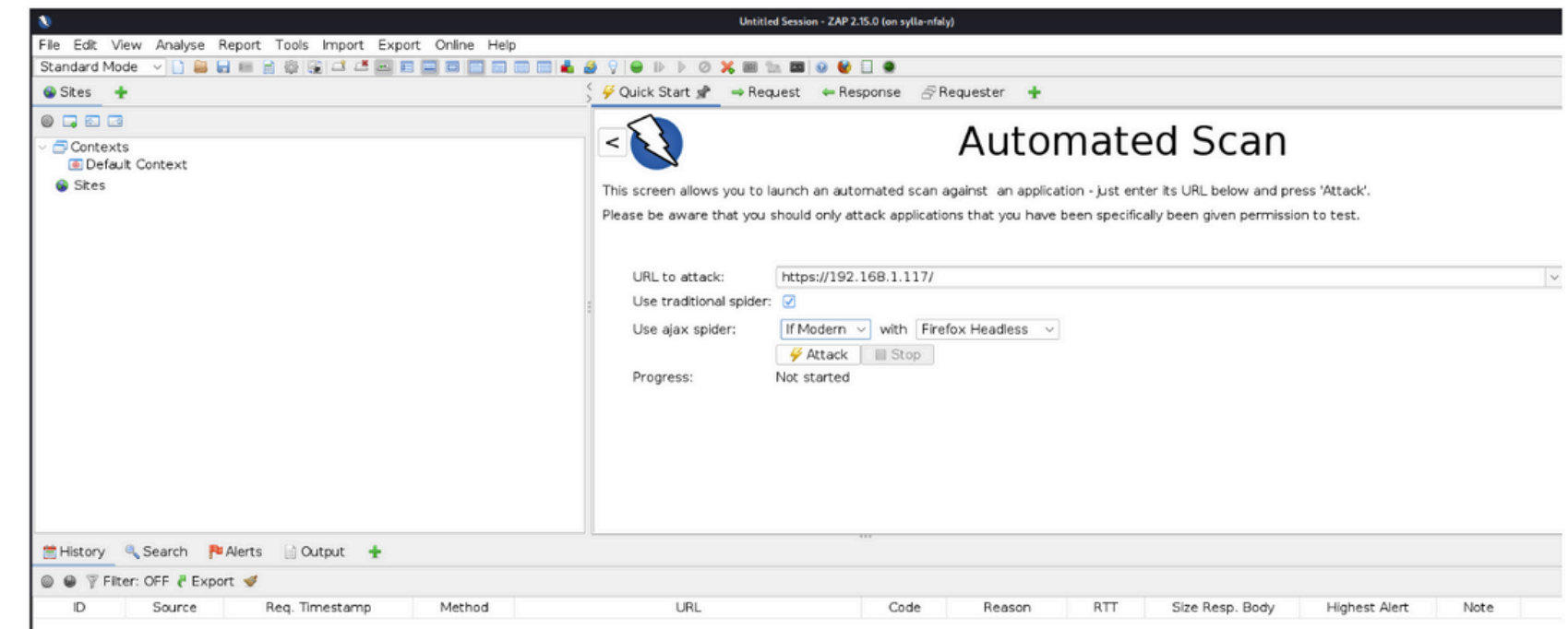
- 🔧 Fonctionnalités clés
 - Scan passif : analyse sans interaction dangereuse (analyse des requêtes/réponses HTTP).
 - Scan actif : simule des attaques pour détecter des vulnérabilités.
 - Détection automatique de failles comme :
 - Injection SQL
 - XSS (Cross-Site Scripting)
 - Configuration serveur faible
 - Authentification mal sécurisée

Outils de test, qualité et CI/CD

OWASP ZAP

Utilisation

1. Lancer l'application web à tester.
2. Ouvrir ZAP et le configurer comme proxy pour votre navigateur.
3. Naviguer dans votre app → ZAP intercepte les requêtes.
4. Lancer un scan actif ou passif.
5. Consulter le rapport de vulnérabilités généré.



Outils de test, qualité et CI/CD

Apache JMeter

Apache JMeter – Tests de performance

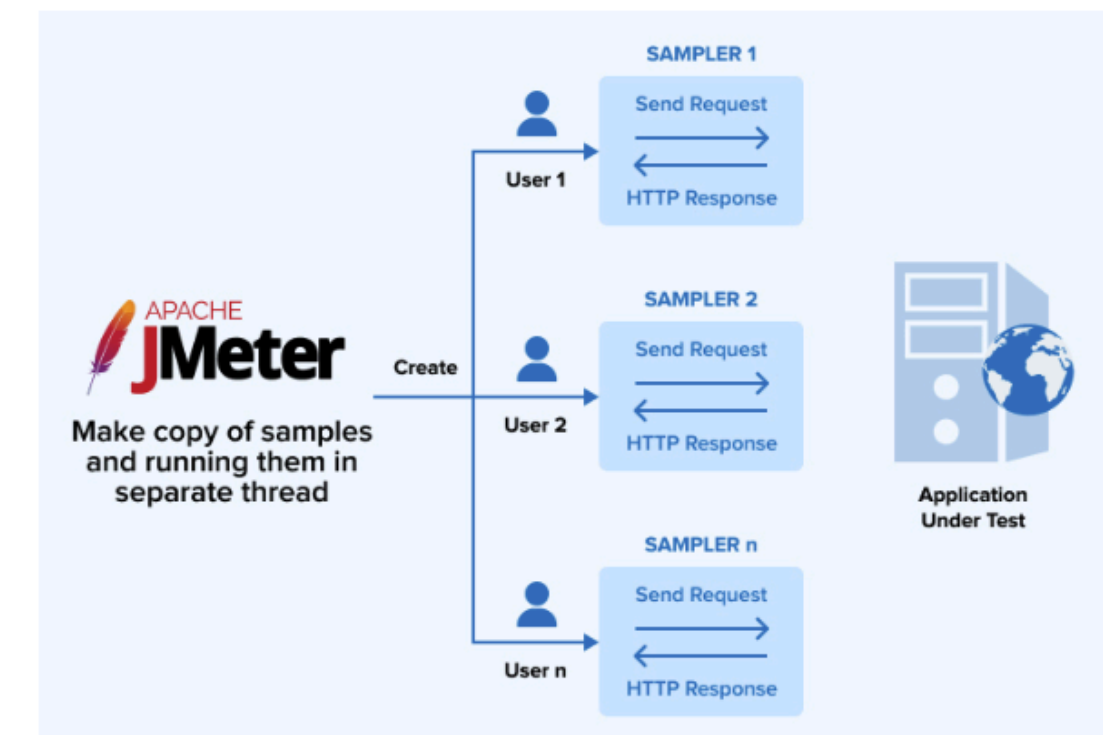
- Apache JMeter est un outil open source développé en Java.
- Créé par la Apache Software Foundation, il est conçu pour :
 - Tester la charge (nombre d'utilisateurs simultanés)
 - Mesurer les performances des applications web et services

Fonctionnalités principales

- Teste des applications Web, FTP, bases de données, APIs, etc.
- Permet d'effectuer :
 - Des tests de performance
 - Des tests fonctionnels
- Simule des utilisateurs simultanés pour évaluer la stabilité et le temps de réponse

Compatibilité multiplateforme

- Fonctionne sur tous les systèmes avec Java :
 - Windows
 - Linux/Unix
 - Mac OS
 - Ubuntu

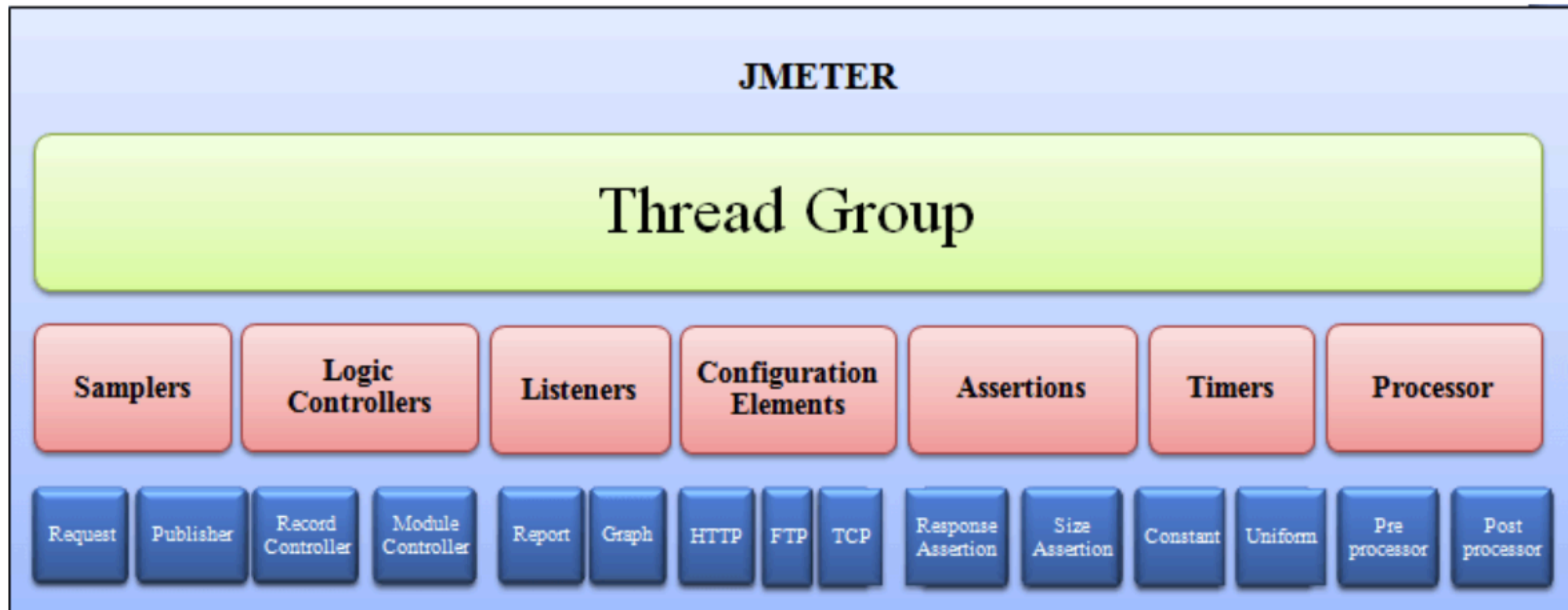


Outils de test, qualité et CI/CD

Apache JMeter

Éléments de JMeter

Dans JMeter, chaque composant utilisé dans un test est appelé élément. Ces éléments permettent de simuler le comportement des utilisateurs, d'exécuter des requêtes, de configurer des paramètres et d'analyser les résultats.



Outils de test, qualité et CI/CD

Apache JMeter



1. Groupe de Threads (Thread Group)

- Simule un ensemble d'utilisateurs virtuels.
- Chaque thread = 1 utilisateur simulé.
- Permet de définir :
 - Le nombre d'utilisateurs
 - La fréquence et la durée d'exécution des tests

2. Échantillonneurs (Samplers)

- Définissent les requêtes envoyées au serveur.
- Types courants :
 - HTTP Request (Web)
 - FTP Request
 - JDBC Request (base de données)
 - SOAP/XML, JSON, etc.

3. Listeners (Écouteurs de résultats)

- Affichent les résultats des tests sous divers formats :
 - View Results Tree : résultats détaillés en HTML
 - Graph Results : graphique de temps de réponse
 - View Results in Table : résumé sous forme de tableau
 - Fichier journal : sortie texte récapitulative

4. Éléments de Configuration

CSV Data Set Config

- Charge dynamiquement des données depuis un fichier CSV
- Idéal pour tester avec plusieurs comptes/utilisateurs

Gestionnaire de Cookies HTTP

- Gère automatiquement les cookies de session (comme un navigateur)

HTTP Request Defaults

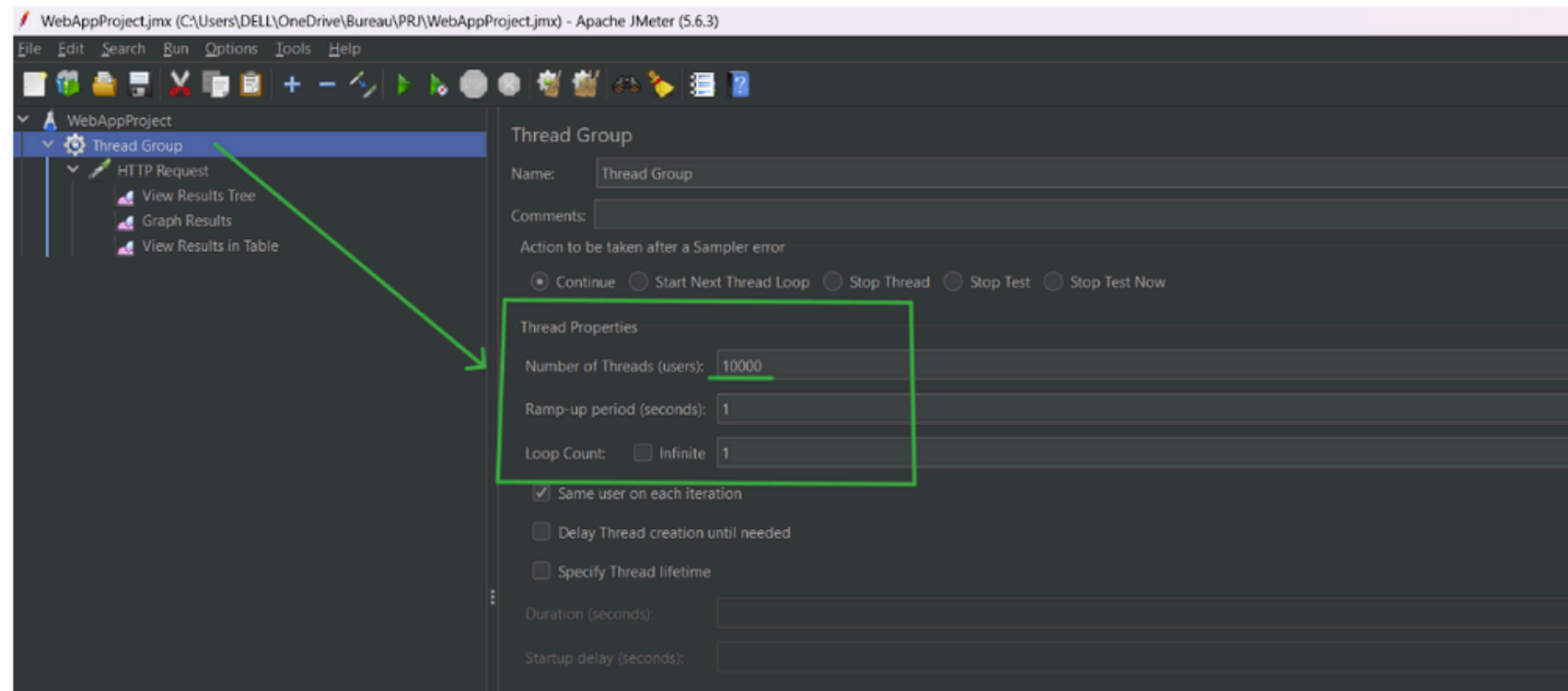
- Définit des valeurs par défaut pour toutes les requêtes HTTP (hôte, port, chemin, etc.)

Outils de test, qualité et CI/CD

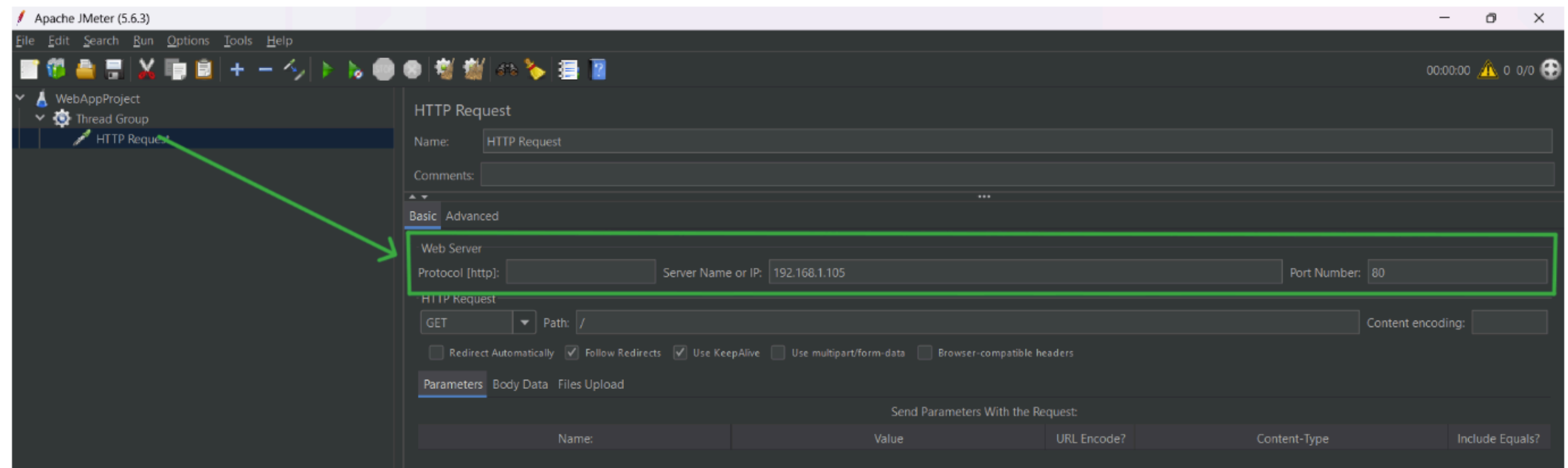
Apache JMeter



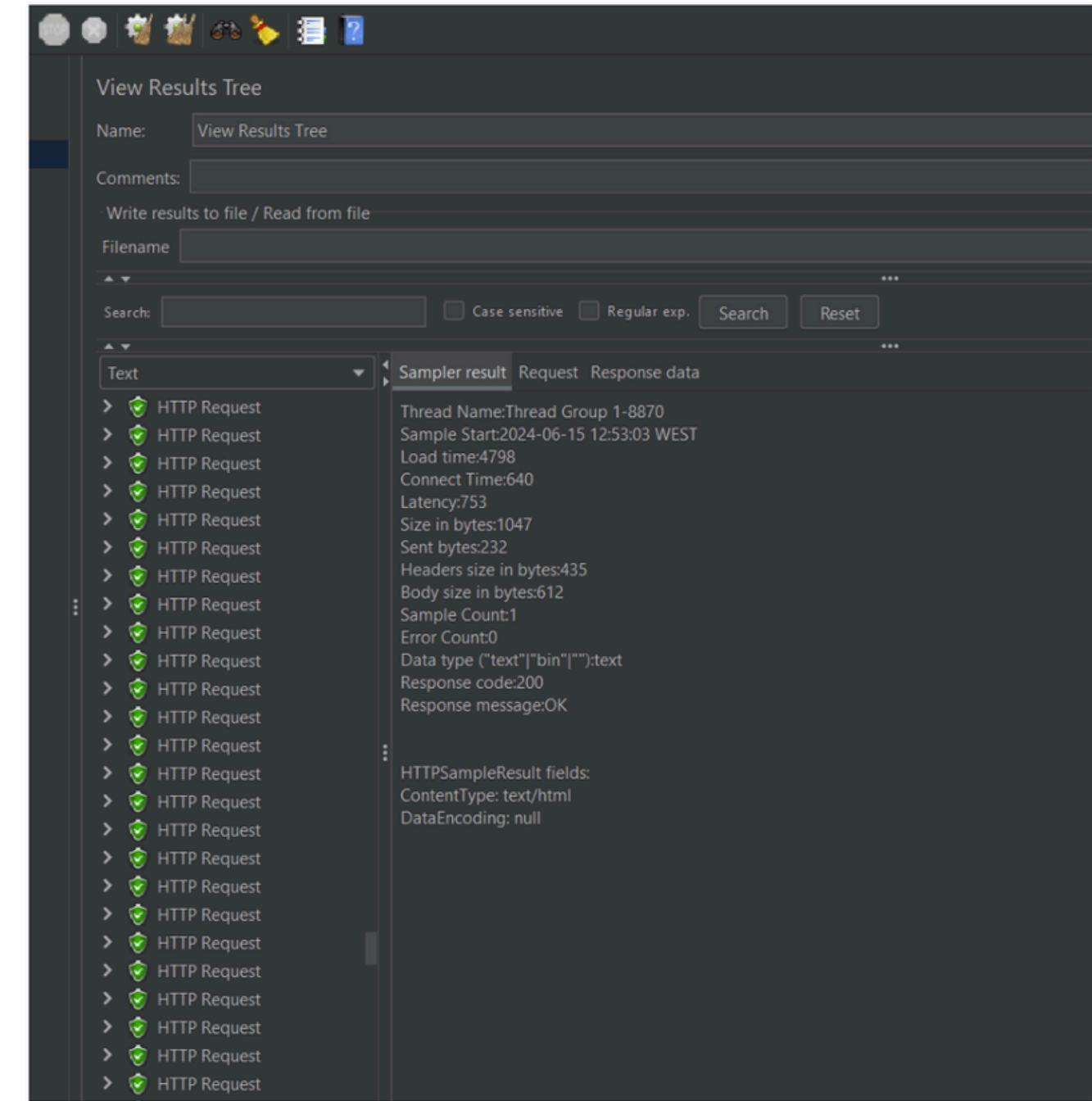
Le thread : On simule 10 000 utilisateurs



Le Sampler Http :



View Results Tree (Arbre de résultats)



View Results in Table (Résultats en tableau) :

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: ☐ Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1045	12-52:52.376	Thread Group 1-9..	HTTP Request	2391		1047	232	938	2
1046	12-52:52.377	Thread Group 1-4..	HTTP Request	2390		1047	232	909	2
1047	12-52:52.369	Thread Group 1-4..	HTTP Request	2398		1047	232	890	2
1048	12-52:52.409	Thread Group 1-4..	HTTP Request	2359	Warning	2795	0	0	2359
1049	12-52:52.408	Thread Group 1-4..	HTTP Request	2360		2795	0	0	2360
1050	12-52:52.410	Thread Group 1-4..	HTTP Request	2358		2795	0	0	2358
1051	12-52:52.408	Thread Group 1-4..	HTTP Request	2360		2795	0	0	2360
1052	12-52:52.410	Thread Group 1-4..	HTTP Request	2358		2795	0	0	2358
1053	12-52:52.410	Thread Group 1-8..	HTTP Request	2358		2795	0	0	2358
1054	12-52:52.410	Thread Group 1-4..	HTTP Request	2358		2795	0	0	2358
1055	12-52:52.409	Thread Group 1-4..	HTTP Request	2359		2795	0	0	2359
1056	12-52:52.409	Thread Group 1-4..	HTTP Request	2359		2795	0	0	2359
1057	12-52:52.414	Thread Group 1-4..	HTTP Request	2354		2795	0	0	2354
1058	12-52:52.407	Thread Group 1-4..	HTTP Request	2362		2795	0	0	2361
1059	12-52:52.409	Thread Group 1-4..	HTTP Request	2360		2795	0	0	2360
1060	12-52:52.409	Thread Group 1-4..	HTTP Request	2360		2795	0	0	2359
1061	12-52:52.410	Thread Group 1-4..	HTTP Request	2359		2795	0	0	2359
1062	12-52:52.409	Thread Group 1-4..	HTTP Request	2359		2795	0	0	2359
1063	12-52:52.408	Thread Group 1-4..	HTTP Request	2359		2795	0	0	2359
1064	12-52:52.408	Thread Group 1-4..	HTTP Request	2361		2795	0	0	2361

Simulation

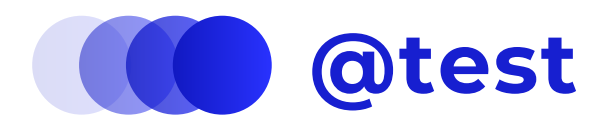
Création et Automatisation des tests avec github actions





Ressources

<https://www.bacancytechnology.com/blog/unit-testing-with-jest-vue-test-utils>
<https://docs.github.com/en/actions/about-github-actions/understanding-github-actions>
https://assets.ctfassets.net/zojzzdop0fzx/VU1dEpLKscEoPc2JHDbkZ/cce207f9051d2b53e3782a4f43eb7b5a/A_Complete_Guide_to_VueJS_by_This_Dot_authored_by_Bilal_Haidar.pdf
<https://arquisoft.github.io/slides/course1920/seminars/SemEn-05-Jest.pdf>
<https://arquisoft.github.io/course2425.html#lectureNotes> <https://jestjs.io/>
<https://docs.cypress.io/app/get-started/why-cypress>
<https://github.com/coderyansolomon/cypress-course.git>



Merci

