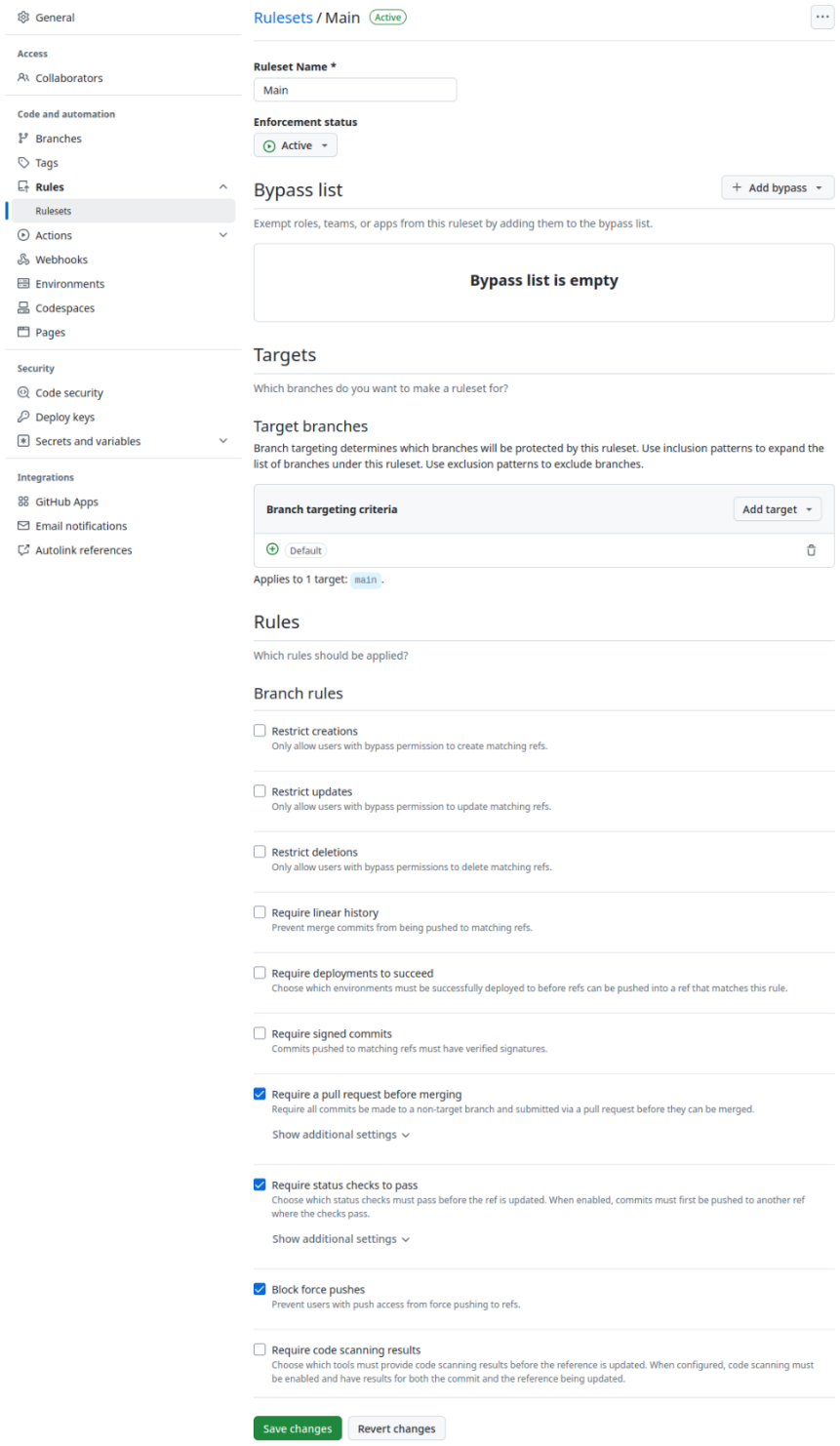
4CITE : Comment mettre en place le Pipeline Git:

Dans les paramètres du repository on doit aller dans Rules > Rulesets > Ajouter



Comme dans le screen, on doit configurer chaque élément :

- Branch targeting criteria : On sélectionne la branche main, ce qui permet de forcer les règles uniquement sur la branche.
- Require a pull request before merging : ce qui force l'utilisateur à créer une pull request pour faire le merge dans le main et de faire une demande de validation d'un autre utilisateur.
    - Require review from Code Owners : La personne qui demande la pull request doit valider son code.
    - Require approval of the most recent reviewable push : Si un nouveau push a lieu sur la branch alors tous les procédés doivent être faient de nouveau.
    - Require conversation resolution before merging : Si la personne qui doit valider ouvre une conversation, elle doit être marquée comme résolue pour valider le merge.
- Require status checks to pass : Ici on met la liste de nos tests, ce qui permet de forcer que les tests soient verts pour que le merge puisse être effectué.
- Block force pushes : Ce qui permet en cas de conflit dans le code, de ne pas forcer le merge.

Comment mettre en place le Lancement du Tests

Dans Actions > new workflow > Node.js > Configure

Il faut ensuite ajouter le fichier suivant

```
name: Node.js CI
on:
push:
branches: [ "main" ]
pull_request:
branches: [ "main" ]

jobs:
build:
runs-on: ubuntu-latest
strategy:
matrix:
node-version: [20.x]
steps:
- name: Checkout Repository
uses: actions/checkout@v4
- name: Start MongoDB Container
run: docker run -d --name mongodb -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=admin -
e MONGO_INITDB_ROOT_PASSWORD=secret -v mongodb_data:/data/db mongo:latest

# ---- Backend ----
- name: Setup Node.js Backend
uses: actions/setup-node@v4
with:
node-version: ${{ matrix.node-version }}
cache: 'npm'
cache-dependency-path: ./Backend/package-lock.json

- name: Install Dependencies Backend
run: npm ci
working-directory: ./Backend

- name: Run Security Audit Backend
run: npm audit --audit-level=critical || echo "Audit completed with issues"
working-directory: ./Backend

- name: Run Tests Backend
run: NODE_OPTIONS='--experimental-vm-modules' npx jest --runInBand
working-directory: ./Backend

# ---- Démarrer le Backend en arrière-plan ----
- name: Start Backend Server
run: npm run dev &
working-directory: ./Backend
```

```
- name: Wait for Backend to be Ready
run: sleep 3

# ---- Frontend ----
- name: Setup Node.js Frontend
uses: actions/setup-node@v4
with:
node-version: ${{ matrix.node-version }}
cache: 'npm'
cache-dependency-path: ./Frontend/package-lock.json

- name: Install Dependencies Frontend
run: npm ci && npm install --save-dev @testing-library/jest-dom && npm i @testing-library/jest-dom
working-directory: ./Frontend

- name: Run Security Audit Frontend
run: npm audit --audit-level=critical || echo "Audit completed with issues"
working-directory: ./Frontend

# - name: Run Tests Frontend
# run: npm test
# working-directory: ./Frontend

# ---- Build & Deploy ----
- name: Build Solution Backend
run: npm run build
working-directory: ./Backend

#- name: Build Solution Frontend
# run: npm run build
# working-directory: ./Frontend

- name: Deploy (Fake Deployment)
run: echo "Deploying solution..."
```

name: Start MongoDB Container run: docker run -d --name mongodb -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=admin MONGO_INITDB_ROOT_PASSWORD=secret -v mongodb_data:/data/db mongo:latest
> Ce qui permet de lancer le container docker.

```
- name: Setup Node.js ${{ matrix.node-version }}
    uses: actions/setup-node@v4
    with:
     node-version: ${{ matrix.node-version }}
     cache: 'npm'
     cache-dependency-path: ./Backend/package-lock.json

   - name: Install Dependencies
    run: npm ci
    working-directory: ./Backend

   - name: Generate package-lock.json if missing
    run: npm i --package-lock-only
    working-directory: ./Backend
```

> Dans cette partie on installe les dépendences

```
- name: Run Security Audit
    run: npm audit --audit-level=critical
    working-directory: ./Backend

   - name: Run Tests
    run: NODE_OPTIONS='--experimental-vm-modules' npx jest --runInBand
    working-directory: ./Backend
```

> Dans cette partie on lance les tests et les audits de sécurité.

```
- name: Build Solution
    run: npm run build
    working-directory: ./Backend

   - name: Deploy (Fake Deployment)
    run: echo "Deploying solution..."
```

> Dans cette partie on lance le build du projet et le déploiement du projet.

Les tests frontend sont mis en commentaire, car nous avons un problème uniquement sur les actions de GitHub avec des dépendances.

Exemple de l’erreur :

```
Run npm run test
npm run test
shell: /usr/bin/bash -e {0}
> frontend@0.0.0 test
> jest
FAIL src/pages/Home.test.tsx
● Test suite failed to run
src/pages/Home.test.tsx:81:51 - error TS2339: Property 'toBeInTheDocument' does not exist on type 'Assertion'.
81 expect(screen.getByTestId('header-mock')).toBeInTheDocument(); .....
```