Workflow Overview

name: BrainBytes Test

on:

push:

branches: [main, development]

pull\_request:

branches: [main, development]

jobs:

test:

name: Run Tests (Node ${{ matrix.node-version }})

runs-on: ubuntu-latest

timeout-minutes: 15

strategy:

matrix:

node-version: [14.x, 16.x, 18.x]

steps:

*# Step 1: Checkout*

- name: Check out repository

uses: actions/checkout@v2

*# Step 2: Setup Node.js*

- name: Set up Node.js ${{ matrix.node-version }}

uses: actions/setup-node@v2

with:

node-version: ${{ matrix.node-version }}

*####################################*

*# FRONTEND*

*####################################*

*# Step 3: Cache frontend dependencies*

- name: Cache frontend node\_modules

uses: actions/cache@v4

with:

path: DevOps/brainbytes-multi-container/frontend/node\_modules

key: ${{ runner.os }}-frontend-v2-${{ matrix.node-version }}-${{ hashFiles('DevOps/brainbytes-multi-container/frontend/package-lock.json') }}

restore-keys: *|*

${{ runner.os }}-frontend-v2-

*# Step 4: Install frontend dependencies*

- name: Install frontend dependencies

working-directory: DevOps/brainbytes-multi-container/frontend

run: *|*

rm -rf node\_modules

if [ -f package-lock.json ]; then

npm install

else

echo "No package-lock.json found, skipping frontend install"

fi

*# Step 5: Run frontend tests with coverage*

- name: Run frontend tests

working-directory: DevOps/brainbytes-multi-container/frontend

run: *|*

if [ -f package.json ]; then

npm run test:coverage || echo "Frontend tests failed"

else

echo "package.json not found in frontend"

fi

*# Step 6: Upload frontend coverage*

- name: Upload frontend coverage

uses: actions/upload-artifact@v4

with:

name: frontend-coverage-${{ matrix.node-version }}

path: DevOps/brainbytes-multi-container/frontend/coverage

*####################################*

*# BACKEND*

*####################################*

*# Step 7: Cache backend dependencies*

- name: Cache backend node\_modules

uses: actions/cache@v4

with:

path: DevOps/brainbytes-multi-container/backend/node\_modules

key: ${{ runner.os }}-backend-v2-${{ matrix.node-version }}-${{ hashFiles('DevOps/brainbytes-multi-container/backend/package-lock.json') }}

restore-keys: *|*

${{ runner.os }}-backend-v2-

*# Step 8: Install backend dependencies*

- name: Install backend dependencies

working-directory: DevOps/brainbytes-multi-container/backend

run: *|*

rm -rf node\_modules

if [ -f package-lock.json ]; then

npm install

else

echo "No package-lock.json found, skipping backend install"

fi

*# Step 9: Run backend tests with coverage*

- name: Run backend tests

working-directory: DevOps/brainbytes-multi-container/backend

run: *|*

if [ -f package.json ]; then

npm run test:coverage || echo "Backend tests failed"

else

echo "package.json not found in backend"

fi

*# Step 10: Upload backend coverage*

- name: Upload backend coverage

uses: actions/upload-artifact@v4

with:

name: backend-coverage-${{ matrix.node-version }}

path: DevOps/brainbytes-multi-container/backend/coverage

1. test.yml - BrainBytes Test

- Purpose: This workflow runs automated tests for both the frontend and backend of the project across multiple Node.js versions (14.x, 16.x, 18.x). It ensures the codebase is functioning correctly and generates test coverage reports.

- Key Features:

- Caches dependencies for faster builds.

- Installs dependencies and runs tests for both frontend and backend.

- Uploads test coverage reports as artifacts.

name: BrainBytes Lint

on:

push:

branches: [main, development]

pull\_request:

branches: [main, development]

jobs:

lint:

name: Code Quality Checks (Node ${{ matrix.node-version }})

runs-on: ubuntu-latest

timeout-minutes: 10

strategy:

matrix:

node-version: [14.x, 16.x, 18.x]

steps:

*# Step 1: Check out the repository*

- name: Check out repository

uses: actions/checkout@v2

*# Step 2: Set up Node.js*

- name: Set up Node.js ${{ matrix.node-version }}

uses: actions/setup-node@v2

with:

node-version: ${{ matrix.node-version }}

*# --- FRONTEND ---*

*# Step 3: Cache frontend dependencies*

- name: Cache frontend dependencies

uses: actions/cache@v3

with:

path: DevOps/brainbytes-multi-container/frontend/node\_modules

key: ${{ runner.os }}-frontend-${{ matrix.node-version }}-${{ hashFiles('DevOps/brainbytes-multi-container/frontend/package-lock.json') }}

restore-keys: *|*

${{ runner.os }}-frontend-${{ matrix.node-version }}-

*# Step 4: Install frontend dependencies*

- name: Install frontend dependencies

working-directory: DevOps/brainbytes-multi-container/frontend

run: npm ci || npm install

*# Step 5: Run ESLint and Prettier check on frontend*

- name: Lint frontend code

working-directory: DevOps/brainbytes-multi-container/frontend

run: *|*

npm run lint:js -- --format json --output-file eslint-report-frontend.json || true

npx prettier --check . || echo "Prettier issues in frontend"

- name: Upload frontend lint report

uses: actions/upload-artifact@v4

with:

name: eslint-report-frontend-${{ matrix.node-version }}

path: DevOps/brainbytes-multi-container/frontend/eslint-report-frontend.json

*# Step 6: Audit frontend dependencies*

- name: Run npm audit on frontend

working-directory: DevOps/brainbytes-multi-container/frontend

run: npm audit --json || true

*# --- BACKEND ---*

*# Step 7: Cache backend dependencies*

- name: Cache backend dependencies

uses: actions/cache@v3

with:

path: DevOps/brainbytes-multi-container/backend/node\_modules

key: ${{ runner.os }}-backend-${{ matrix.node-version }}-${{ hashFiles('DevOps/brainbytes-multi-container/backend/package-lock.json') }}

restore-keys: *|*

${{ runner.os }}-backend-${{ matrix.node-version }}-

*# Step 8: Install backend dependencies*

- name: Install backend dependencies

working-directory: DevOps/brainbytes-multi-container/backend

run: npm ci || npm install

*# Step 9: Run ESLint and Prettier check on backend*

- name: Lint backend code

working-directory: DevOps/brainbytes-multi-container/backend

run: *|*

npm run lint:js -- --format json --output-file eslint-report-backend.json || true

npx prettier --check . || echo "Prettier issues in backend"

- name: Upload backend lint report

uses: actions/upload-artifact@v4

with:

name: eslint-report-backend-${{ matrix.node-version }}

path: DevOps/brainbytes-multi-container/backend/eslint-report-backend.json

*# Step 10: Audit backend dependencies*

- name: Run npm audit on backend

working-directory: DevOps/brainbytes-multi-container/backend

run: npm audit --json || true

2. lint.yml - BrainBytes Lint

- Purpose: This workflow performs code quality checks using ESLint and Prettier for both the frontend and backend. It also audits dependencies for known vulnerabilities.

- Key Features:

- Caches dependencies for faster builds.

- Runs ESLint and Prettier checks.

- Uploads lint reports as artifacts.

- Runs `npm audit` to identify security issues in dependencies.

name: BrainBytes Docker Build

on:

push:

branches: [main, development]

paths-ignore:

- '\*\*.md'

- 'docs/\*\*'

workflow\_dispatch:

jobs:

build:

name: Build Docker Images

runs-on: ubuntu-latest

timeout-minutes: 15

steps:

*# Step 1: Check out the repository*

- name: Check out repository

uses: actions/checkout@v3

*# Step 2: Set up Docker Buildx*

- name: Set up Docker Buildx

uses: docker/setup-buildx-action@v3

*# Step 3: Login to GitHub Container Registry (optional for push)*

*# - name: Log in to GHCR*

*# uses: docker/login-action@v3*

*# with:*

*# registry: ghcr.io*

*# username: ${{ github.actor }}*

*# password: ${{ secrets.GITHUB\_TOKEN }}*

*# Step 4: Build frontend image*

- name: Build frontend image

uses: docker/build-push-action@v4

with:

context: ./DevOps/brainbytes-multi-container/frontend

file: ./DevOps/brainbytes-multi-container/frontend/Dockerfile

push: false

tags: brainbytes/frontend:latest

cache-from: type=gha

cache-to: type=gha,mode=max

outputs: type=docker,dest=/tmp/frontend-image.tar

*# Step 5: Build backend image*

- name: Build backend image

uses: docker/build-push-action@v4

with:

context: ./DevOps/brainbytes-multi-container/backend

file: ./DevOps/brainbytes-multi-container/backend/Dockerfile

push: false

tags: brainbytes/backend:latest

cache-from: type=gha

cache-to: type=gha,mode=max

outputs: type=docker,dest=/tmp/backend-image.tar

*# Step 6: Upload built image artifacts*

- name: Upload Docker images

uses: actions/upload-artifact@v4

with:

name: docker-images

path: *|*

/tmp/frontend-image.tar

/tmp/backend-image.tar

*# Step 7: Install Docker Compose (if needed)*

- name: Install Docker Compose

run: *|*

sudo curl -L "https://github.com/docker/compose/releases/download/v2.17.2/docker-compose-linux-x86\_64" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version

*# Step 8: Test Docker Compose setup*

- name: Test Docker Compose

working-directory: ./DevOps/brainbytes-multi-container

run: *|*

docker-compose up -d

sleep 10

docker-compose ps

docker-compose down

3. docker-build.yml - BrainBytes Docker Build

- Purpose: This workflow builds Docker images for the frontend and backend services. It ensures that the Docker images are ready for deployment.

- Key Features:

- Builds Docker images for both frontend and backend.

- Caches Docker layers for faster builds.

- Uploads built Docker images as artifacts.

- Tests the Docker Compose setup to verify the multi-container application works as expected.

name: BrainBytes Deploy

on:

workflow\_run:

workflows:

- BrainBytes Lint

- BrainBytes Test

- BrainBytes Docker Build

types:

- completed

jobs:

deploy:

if: ${{ github.event.workflow\_run.conclusion == 'success' }}

runs-on: ubuntu-latest

timeout-minutes: 20

steps:

- name: Checkout repository

uses: actions/checkout@v4

- name: Set up Docker Buildx

uses: docker/setup-buildx-action@v3

- name: Log in to Docker Hub

uses: docker/login-action@v3

with:

username: ${{ secrets.DOCKER\_USERNAME }}

password: ${{ secrets.DOCKER\_PASSWORD }}

- name: Push frontend image

uses: docker/build-push-action@v4

with:

context: ./DevOps/brainbytes-multi-container/frontend

file: ./DevOps/brainbytes-multi-container/frontend/Dockerfile

push: true

tags: d1tt0/brainbytes-frontend:latest

- name: Push backend image

uses: docker/build-push-action@v4

with:

context: ./DevOps/brainbytes-multi-container/backend

file: ./DevOps/brainbytes-multi-container/backend/Dockerfile

push: true

tags: d1tt0/brainbytes-backend:latest

*# Optional: Deploy via SSH*

*# - name: SSH to server and redeploy*

*# uses: appleboy/ssh-action@v1.0.0*

*# with:*

*# host: ${{ secrets.SERVER\_HOST }}*

*# username: ${{ secrets.SERVER\_USER }}*

*# key: ${{ secrets.SERVER\_SSH\_KEY }}*

*# script: |*

*# docker pull d1tt0/brainbytes-frontend:latest*

*# docker pull d1tt0/brainbytes-backend:latest*

*# docker-compose down*

*# docker-compose up -d --remove-orphans*

4. deploy.yml - BrainBytes Deploy

- Purpose: This workflow deploys the application to a Docker registry (e.g., Docker Hub) after successful completion of the `Lint`, `Test`, and `Docker Build` workflows.

- Key Features:

- Pushes Docker images for the frontend and backend to Docker Hub.

- (Optional) Includes an SSH deployment step for remote servers.

## How to Run Workflows Manually

1. Triggering Workflows via GitHub UI :

- Navigate to the Actions tab in your repository.

- Select the desired workflow (e.g., `BrainBytes Test`, `BrainBytes Lint`, etc.).

- Click the "Run workflow" button.

- If applicable, provide any required input parameters and confirm.

2. Triggering docker-build.yml Manually :

- This workflow supports manual triggering via the `workflow\_dispatch` event.

- Follow the steps above to trigger it manually.

Workflow Status Badges

You can add the following badges to your repository's README to display the status of each workflow:

```markdown

Test Workflow

![BrainBytes Test](https://github.com/jayceebasa/DevOps---S3101---B4lagbagan/actions/workflows/test.yml/badge.svg)

Lint Workflow

![BrainBytes Lint](https://github.com/jayceebasa/DevOps---S3101---B4lagbagan/actions/workflows/lint.yml/badge.svg)

Docker Build Workflow

![BrainBytes Docker Build](https://github.com/jayceebasa/DevOps---S3101---B4lagbagan/actions/workflows/docker-build.yml/badge.svg)

Deploy Workflow

![BrainBytes Deploy](https://github.com/jayceebasa/DevOps---S3101---B4lagbagan/actions/workflows/deploy.yml/badge.svg)

```

- Green Badge : Workflow completed successfully.

- Red Badge : Workflow failed.

- Yellow Badge : Workflow is in progress or skipped.

Replace `<OWNER>` and `<REPO>` with your GitHub username and repository name.

---

Troubleshooting Instructions

Common Issues and Fixes

1. Dependency Caching Issues :

- Error : "Cache not found for input keys."

- Fix : Ensure the `package-lock.json` file exists in the specified path. If missing, regenerate it using `npm install`.

2. Test Failures :

- Error : "Frontend/Backend tests failed."

- Fix : Check the test logs in the Actions tab. Ensure all test dependencies are installed and the test scripts are correctly defined in `package.json`.

3. Docker Build Failures :

- Error : "Failed to build Docker image."

- Fix : Verify the `Dockerfile` syntax and ensure all required files are present in the build context.

4. Deployment Issues :

- Error : "Authentication failed for Docker Hub."

- Fix : Ensure the `DOCKER\_USERNAME` and `DOCKER\_PASSWORD` secrets are correctly configured in the repository settings.

5. SSH Deployment Issues (Optional Step in deploy.yml):

- Error : "Permission denied."

- Fix : Verify the SSH key and server credentials in the repository secrets.

For further assistance, refer to the GitHub Actions [documentation](https://docs.github.com/en/actions).

jayceebasa: can you export that to a docx file

GitHub Copilot: I cannot directly create or export a `.docx` file. However, I can guide you on how to do it:

1. Copy the YAML content provided above.

2. Open Microsoft Word or any word processor that supports `.docx` files.

3. Paste the content into the document.

4. Save the file as a `.docx` format.

If you need help formatting the content in Word (e.g., adding headings or styling), let me know!