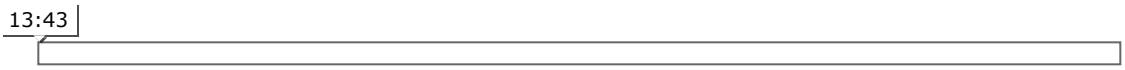
Watch: ⚡ **Integramos el testing en nuestra pipeline**



Puedes encontrar el código de este vídeo en GitHub.

Los tests no nos van a servir de mucho si no nos aseguramos de ejecutarlos periódicamente. La mejor forma de asegurar esto es integrarlos en nuestro proceso de desarrollo con herramientas de integración continua.

En este vídeo nos centraremos en GitHub Actions, pero hay muchas alternativas como las pipelines de GitLab, CircleCI, etc.

Para crear un workflow que ejecute los tests por cada push, añadiremos un fichero yaml en la carpeta `.github/workflows`:

```
name: test
```

```
on: push

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v2

      - name: Cache node modules
        uses: actions/cache@v2
        env:
          cache-name: cache-node-modules
        with:
          path: ~/.npm
          key: ${{ runner.os }}-build-${{ env.cache-name }}-${{
hashFiles('**/package-lock.json') }}
          restore-keys: |
            ${{ runner.os }}-build-${{ env.cache-name }}-
            ${{ runner.os }}-build-
            ${{ runner.os }}-

      - name: Install Dependencies
        run: npm install
        working-directory: ./91-ci # solo añadir si no tenemos
el proyecto en la raíz

      - name: Test
        run: npm test
        working-directory: ./91-ci # solo añadir si no tenemos
el proyecto en la raíz
```

También tenemos la opción de filtrar por rama, o ejecutarlos en otros eventos como pull request. Tienes una lista de todos los triggers en la documentación de GitHub Actions.

También podemos forzarnos a ejecutar los tests en local con Husky, un paquete que nos permite ejecutar ciertos scripts antes de hacer commit o push. Nosotros recomendamos ejecutarlos antes de hacer push y no commit, para no bloquearnos si necesitamos hacer commit de un paso intermedio en el que fallan los tests.