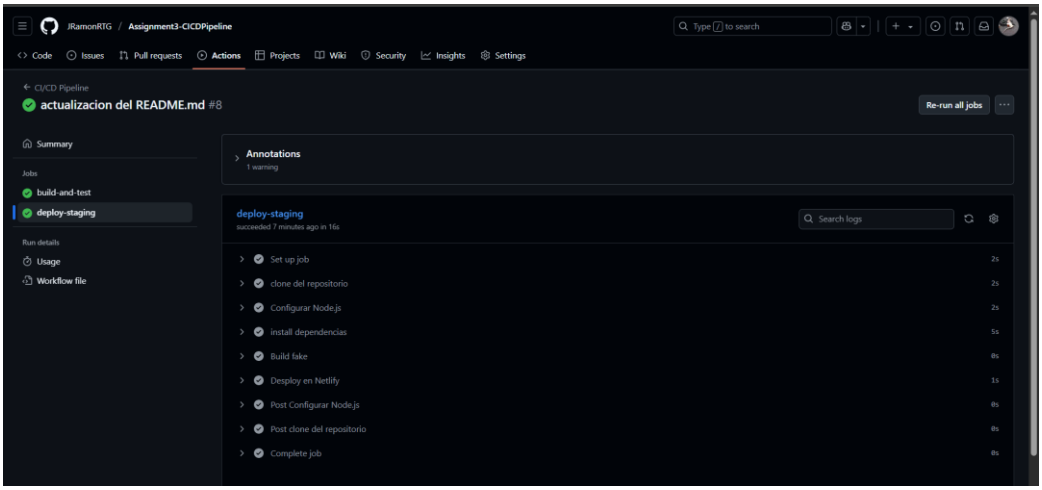
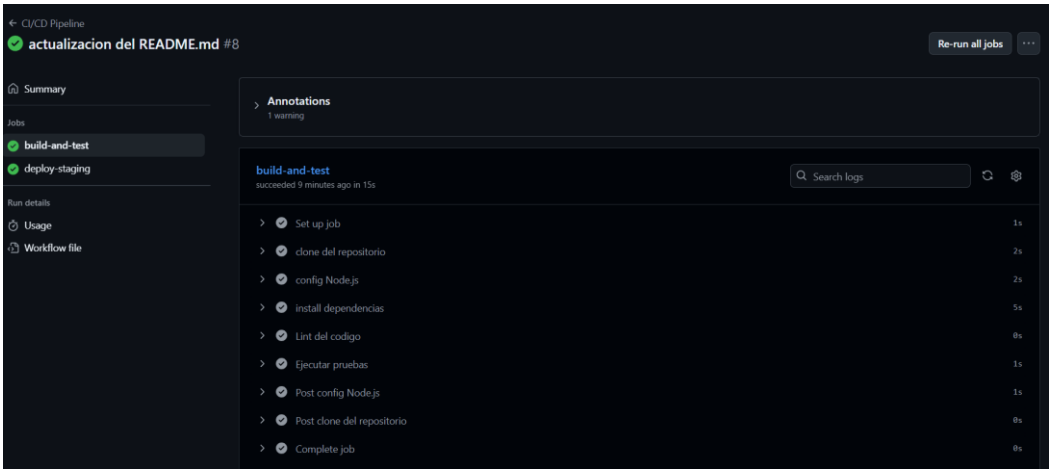
Assignment 3 – Build and Evaluate Your Own CI/CD Pipeline

Pipeline actual CI/CD:

Paso	Acción	Descripción
1	`checkout`	Clona el repositorio del proyecto
2	`setup-node`	Configura Node.js v18
3	`npm ci`	Instala dependencias basadas en `package-lock.json`
4	`npm run lint`	Ejecuta ESLint (`eslint.config.js`)
5	`npm test`	Ejecuta pruebas unitarias con Jest
6	Deploy en Netlify	Si todo pasa, despliega en Netlify (staging) usando secrets seguros

Pipeline en ejecución:





Reflexión Personal

- **What was easiest/hardest?**
  Lo más fácil fue conectar GitHub con Netlify para despliegue automático, ya que fue crear la cuenta, vincular con git, crear el proyecto para luego configurarlo para el repo y generar los tokens. Lo más difícil diría que fue resolver errores del pipeline, ya que algunos eran causados por cambios en ESLint 9 teniendo una migración a eslint.config.js así que debía de configurar mi código o el archivo, en algunos casos tuve varios errores que eran incompatibilidades o falta de instalaciones, como pueden ser Jest como dependencia.

- **How did this pipeline improve the delivery process?**
  Ahora que lo veo un poco mejor podemos detectar errores antes del despliegue y automatiza el proceso sin intervención manual directamente. Esto hace que se reduzcan los riesgos por errores imprevistos o no detectados del todo de forma manual y mejora la velocidad de entrega haciendo el deploy automático, siendo justo parte con las prácticas ágiles.

- **How would you expand it for a full product team?**

  Posiblemente lo siguiente seria tomar en cuenta los diferentes roles o divisiones que se tendrá el team, en dependencia de eso y del numero de personas tal vez podría ser en Frontend, Backend (si se expande a un proyecto donde ya la app guarde el clima o algo por estilo imagino), estos siendo del grupo dev y aparte los grupos de test y production.

  Esto para que cada uno trabaje en su parte, dejando de lado también se tomaría en cuenta la implementación de pruebas de integración, cobertura, entornos separados por si algo falla y no afecte a todo, y revisiones automáticas con despliegue que se ejecuten solo a partir de ciertas condiciones y no solo porque paso las pruebas automáticas (esto por si se necesita una verificación final manual por alguna razón como un cambio que solo se pondrá disponible a cierta hora o algún tipo de evento)?

URL del repo de Github:

https://github.com/JRamonRTG/Assignment3-CICDPipeline.git