

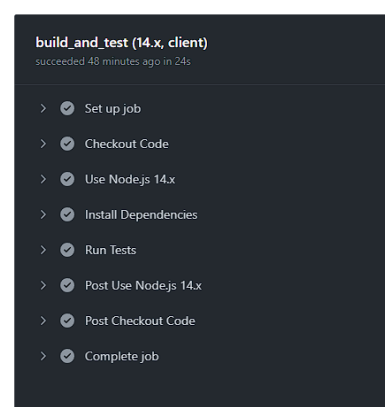
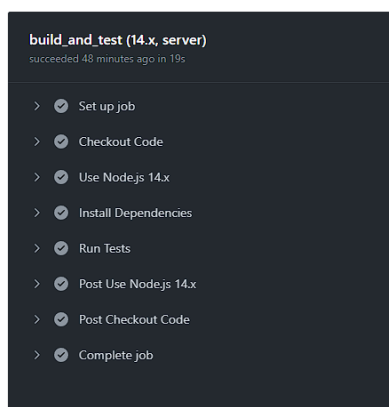
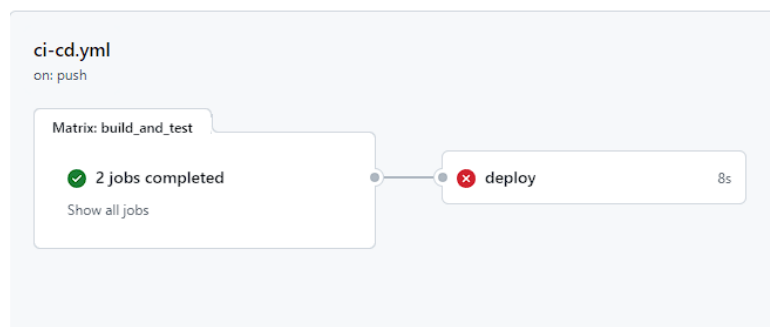
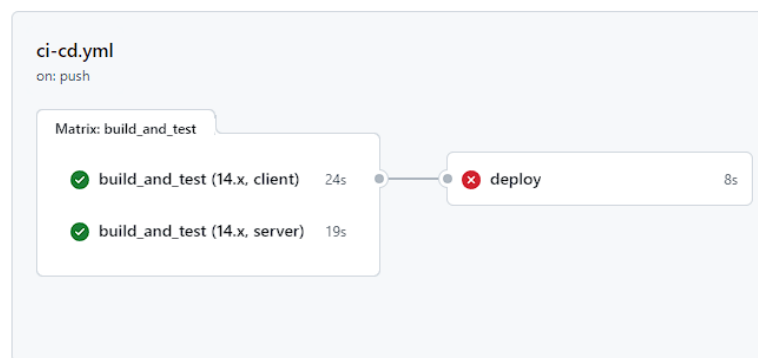
CI / CD con Actions de Github

Para el CI / CD, he considerado trabajar con los Actions de Github por tema de tiempo, funcionalidad más joven, ante el requerimiento de hacerlo con Jenkins, explico las ventajas y desventajas más abajo.

Se construye el pipeline con las etapas de **build_and_test** y **deploy** de forma general, pero se puede realizar mejoras, como la construcción de estas dos etapas por cada aplicación, esto es, el frontend y el backend de forma independiente, que sería lo adecuado.


Los Actions son construidos en archivos yaml, que desarrollan cada paso del flujo de trabajo deseado. Además, también existe un Marketplace de Actions que se pueden considerar para agregarlos al proyecto en función de la necesidad que se tenga.


Finalmente, se deja para probar, el ingreso de las credenciales de una cuenta en AWS para la ejecución del deploy.




deploy

failed 48 minutes ago in 8s


>  Set up job

>  Checkout Code


▼  Configure AWS Credentials

1 ▶ Run aws-actions/configure-aws-credentials@v1

5 **Error:** Credentials could not be loaded, please check your action inputs: Could not load credentials from any providers

 Deploy to AWS

>  Post Configure AWS Credentials

>  Post Checkout Code

>  Complete job

Code

Blame

49 lines (39 loc) · 1.18 KB



Code 55% faster with GitHub Copilot

```
1   name: CI/CD Pipeline
2
3   on: [push]
4
5   jobs:
6     build_and_test:
7       runs-on: ubuntu-latest
8       strategy:
9         matrix:
10          node-version: [14.x]
11          directory: ['client', 'server']
12
13       steps:
14         - name: Checkout Code
15           uses: actions/checkout@v2
16
17         - name: Use Node.js ${ matrix.node-version }
18           uses: actions/setup-node@v2
19           with:
20             node-version: ${ matrix.node-version }
21
22         - name: Install Dependencies
23           run: npm install
24           working-directory: ${ matrix.directory }
25
26         - name: Run Tests
27           run: npm test
28           working-directory: ${ matrix.directory }
29
30     deploy:
31       needs: build_and_test
32       runs-on: ubuntu-latest
33
34       steps:
35         - name: Checkout Code
36           uses: actions/checkout@v2
37
38         - name: Configure AWS Credentials
39           uses: aws-actions/configure-aws-credentials@v1
40           with:
41             aws-access-key-id: ${ secrets.AWS_ACCESS_KEY_ID }
42             aws-secret-access-key: ${ secrets.AWS_SECRET_ACCESS_KEY }
43             aws-region: us-west-2 # or your preferred region
44
45         # Add your steps to deploy to AWS
46         - name: Deploy to AWS
47           run: |
48             cd server
49             # Add your AWS deployment commands here
```

Jenkins y GitHub Actions son herramientas que proporcionan integración continua y entrega continua (CI/CD). A continuación, describiré las diferencias, similitudes, pros y contras de cada una.

Jenkins:

Pros:

1. Madurez: Jenkins ha existido por mucho tiempo y tiene una comunidad robusta y activa.
2. Extensibilidad: Con más de 1,500 plugins, Jenkins es extremadamente extensible.
3. Flexible: Puede configurarse para casi cualquier flujo de trabajo de CI/CD.
4. Multiplataforma: Puede instalarse en máquinas con diferentes sistemas operativos.

Contras:

1. Curva de aprendizaje: Configurar y optimizar Jenkins puede ser complicado.
2. Mantenimiento: Requiere mantenimiento constante y manual, especialmente con actualizaciones.
3. Interfaz de usuario: Aunque es funcional, la interfaz de usuario no es tan moderna o intuitiva.

GitHub Actions:

Pros:

1. Integración con GitHub: Ofrece una integración perfecta con los repositorios de GitHub.
2. Facilidad de uso: Configuración sencilla con archivos YAML y una interfaz de usuario clara y moderna.
3. Ambiente de ejecución: Proporciona ambientes de ejecución predefinidos y personalizables para los flujos de trabajo.
4. Escalabilidad: Automáticamente escalable y sin necesidad de mantenimiento.

Contras:

1. Costo: Puede ser costoso para proyectos grandes con muchas ejecuciones de CI/CD.
2. Límites de uso: Existen límites en los minutos de ejecución y otros recursos.
3. Menos Plugins: Aunque crece rápidamente, tiene menos plugins y extensiones comparado con Jenkins.

Tabla Comparativa:

Característica	Jenkins	GitHub Actions
Integración con GitHub	Necesita plugins y configuración adicional	Integración nativa
Facilidad de Uso	Curva de aprendizaje más pronunciada	Más fácil y directo
Extensibilidad	Más de 1500 plugins	Menos plugins, pero en crecimiento
Configuración	Requiere instalación y configuración manual	Configuración basada en YAML
Mantenimiento	Requiere mantenimiento constante	Bajo o nulo mantenimiento
Costo	Principalmente auto-hospedado	Precio basado en uso
Interfaz de Usuario	Menos intuitiva y moderna	Interfaz moderna y limpia
Madurez y Comunidad	Comunidad amplia y madura	Comunidad creciente y activa
Multiplataforma	Puede instalarse en diferentes SO	Servicio en la nube

Conclusión:

La elección entre Jenkins y GitHub Actions dependerá de tus necesidades específicas, recursos y preferencias. Para proyectos nuevos, especialmente aquellos alojados en GitHub, GitHub Actions puede ser una opción más simple y directa. Para organizaciones con infraestructuras existentes y complejas, Jenkins puede ofrecer la flexibilidad y control necesarios.