UNIVERSIDAD DEL VALLE DE GUATEMALA

Tarea 5. Integración Contínua/DevOps

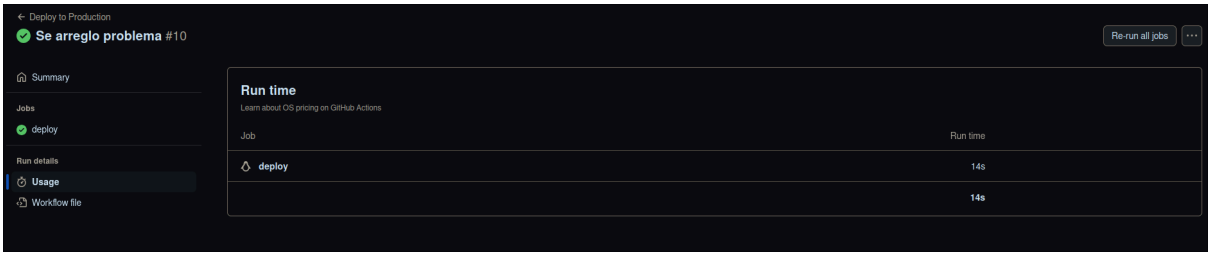
BRANDON JAVIER REYES MORALES - 22992
CARLOS ALBERTO VALLADARES GUERRA - 221164
GUSTAVO ADOLFO CRUZ BARDALES- 22779
PAULA REBECA BARILLAS ALVAREZ - 22764
RODRIGO ALFONSO MANSILLA DUBÓN - 22611

Catedrático: Lynette García Pérez

Ingeniería de Software
Guatemala, 2024

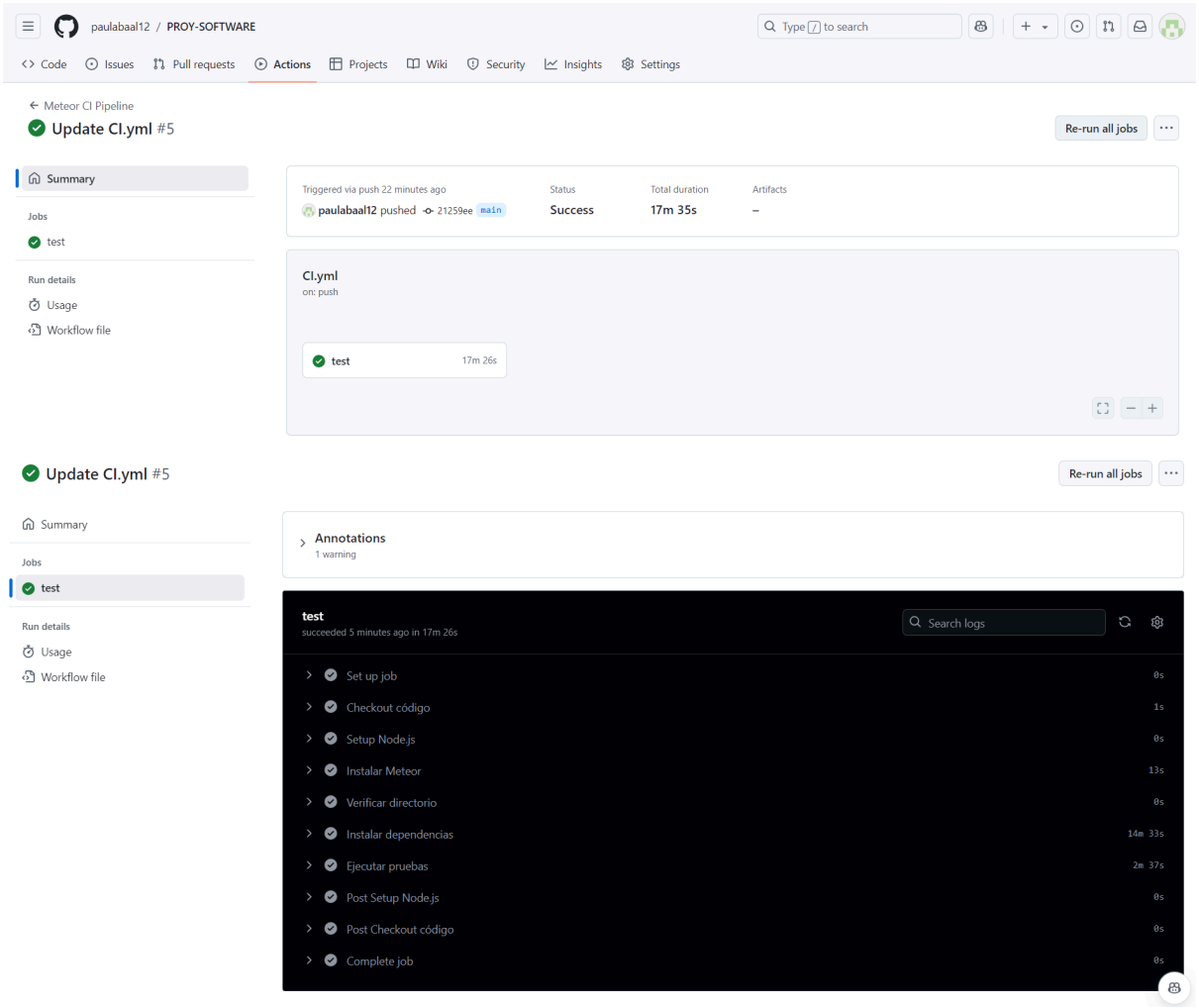
Evidencia de ejecución exitosa:

Despliegue Continuo (CD)




```
STEP 10/17: RUN npm install -g npm@latest --legacy-peer-deps
--> Using cache 403ff4c64fa2b9b11323f667baa8f2d6cc5dcecd93db44cf1fa5e04d895033c7
--> 403ff4c64fa2
STEP 11/17: RUN mkdir -p /usr/src/app/.meteor/local && chown -R meteoruser:meteoruser /usr
--> Using cache 2287e96846c52bf197faef1950ee7c35ffa980bf62ce69af9104af175ce0c592
--> 2287e96846c5
STEP 12/17: USER meteoruser
--> Using cache d1f93b393274f0b7948b609d874b5102f9c51e7a707bf9df391eb471b48918e2
--> d1f93b393274
STEP 13/17: RUN meteor update
--> Using cache 7d125d274237993b5385784db4f9cfc8edd5f31b1561a0184e0f3a0a6a83e558
--> 7d125d274237
STEP 14/17: RUN meteor npm install --legacy-peer-deps
--> Using cache edfa044b99db36060ead55b1261e89cbf87dc827645103af61ad09136222c03b
--> edfa044b99db
STEP 15/17: COPY . .
--> c9dab16764c2
STEP 16/17: EXPOSE 3000
--> e20a61b45a98
STEP 17/17: CMD [ "npm", "start"]
COMMIT meteor_app
--> 2f0f34d6969c
Successfully tagged localhost/meteor_app:latest
2f0f34d6969ce234646cd25083cb2f2a438f7c0fef6bc7cbf1ac68c3d8a274e5
Reconstruyendo DBSTEP 1/3: FROM postgres:16-alpine3.20
STEP 2/3: COPY ./schema.sql /docker-entrypoint-initdb.d/
--> Using cache 7d5189914ade2f98ebd298e622f1c780b73737c1697968e78304778968ce7202
--> 7d5189914ade
STEP 3/3: EXPOSE 5432
--> Using cache a9ab1ba39f582b9e45be1a7debba8f59b873e4ed4e0579147d3591a3b5d13283
COMMIT meteor_postgresdb
--> a9ab1ba39f58
Successfully tagged localhost/meteor_postgresdb:latest
a9ab1ba39f582b9e45be1a7debba8f59b873e4ed4e0579147d3591a3b5d13283
Levantando contenedores7a4f1fff229d8eb6440d2be87a8c3522b8d475eb2f6c070d9fc4f50ecc048678
0ca4b12bd8bb22107674914bce5d03ca62f9170465506442c28b017ce7e3c6c5
```

Configuración de Integración Continua (CI)



Documentación

El pipeline automatiza el proceso de despliegue continuo e integración continua. Cada vez que se realiza un push al branch principal, el pipeline se encarga de ejecutar los tests y desplegar automáticamente la última versión de la aplicación en el servidor de producción.

Se utilizan las siguientes tecnologías:

- **GitHub Actions**: Plataforma de automatización para CI/CD.
- **Mocha/Chai**: Frameworks de pruebas unitarias para JavaScript.
- **Meteor**: Framework de JavaScript para aplicaciones web.
- **SSH**: Protocolo para la conexión segura al servidor de producción.
- **GitHub Secrets**: Almacenamiento seguro de credenciales y claves privadas.

Flujo del pipeline

- CI

    - Se ejecutan pruebas unitarias cada vez que se realiza un push al branch principal o se abre un pull request.
    - Asegura que los cambios en el código no introduzcan errores y mantienen la calidad del proyecto.
- CD
    - Despliega automáticamente la aplicación al servidor de producción cada vez que se realiza un *push* al branch principal.
    - Utiliza SSH para conectarse al servidor y ejecutar comandos que actualizan la aplicación.

Archivo de configuración CI

```yaml
name: Meteor CI Pipeline

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main

jobs:
  test:
    runs-on: ubuntu-latest

    defaults:
      run:
        working-directory: ./Scrum/software

    steps:
      - name: Checkout código
        uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '20.17.0'

      - name: Instalar Meteor
        run: |
          curl https://install.meteor.com/ | sh
```

```
    - name: Verificar directorio
      run: |
        pwd
        ls -la

    - name: Instalar dependencias
      run: meteor npm install

    - name: Ejecutar pruebas
      run: meteor npm test
```

Checkout del Código:

- `actions/checkout@v3` para obtener el código fuente del repositorio.

Configuración de Node.js:

- Configura la versión de Node.js requerida (`20.17.0`) utilizando `actions/setup-node@v3`.

Instalación de Meteor:

- Instala Meteor mediante un script proporcionado por Meteor.

Verificación del Directorio:

- Ejecuta comandos `pwd` y `ls -la` para verificar que el directorio de trabajo es correcto y que los archivos están presentes.

Instalación de Dependencias:

- Instala las dependencias del proyecto usando `meteor npm install`.

Ejecución de Pruebas:

- Ejecuta las pruebas unitarias definidas en el proyecto con `meteor npm test`.

Archivo de configuración CI

```
name: Deploy to Production

on:
```

```
  push:
    branches:
      - main

jobs:
  deploy:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v2

      - name: Set up SSH
        uses: webfactory/ssh-agent@v0.5.3
        with:
          ssh-private-key: ${{ secrets.SSH_PRIVATE_KEY }}

      - name: Deploy via SSH
        run: |
          ssh -o StrictHostKeyChecking=no -i /home/runner/.ssh/id_rsa
gustavo@190.148.223.41 << 'EOF'
            cd /home/gustavo/PROY-SOFTWARE/Scrum/software/
            git pull origin main
            bash rebuild.sh
          EOF
```

Checkout del Repositorio:

- Utiliza `actions/checkout@v2` para obtener el código fuente del repositorio en el entorno de ejecución.

Configuración de SSH:

- Configura el agente SSH usando `webfactory/ssh-agent@v0.5.3` y carga la clave privada almacenada en los **GitHub Secrets** (`SSH_PRIVATE_KEY`).

Despliegue vía SSH:

- Conecta al servidor de producción mediante SSH.
- Navega al directorio de la aplicación.
- Ejecuta `git pull origin main` para actualizar el código con los últimos cambios del branch principal.

- Ejecuta un script `rebuild.sh` para reconstruir y reiniciar la aplicación si es necesario.

Decisiones de Diseño y Configuración

Elección de GitHub Actions

Se decidió el uso de **GitHub Actions** por su integración nativa con GitHub,facilitando la configuración y el mantenimiento de los pipelines de CI/CD y por su capacidad de personalización del flujo del pipeline.

Framework de Pruebas Automatizadas

Se optó **Mocha/Chai** por su robustez y facilidad de uso en proyectos JavaScript.

Estructura del Archivo YAML

Se divide la configuración en dos archivos YAML separados (`ci.yml` y `cd.yml`) para mantener una separación de responsabilidades entre la integración continua y el despliegue continuo.

Uso de GitHub Secrets

Se implementa **GitHub Secrets** para manejar de manera segura las credenciales sensibles, como la clave privada SSH.