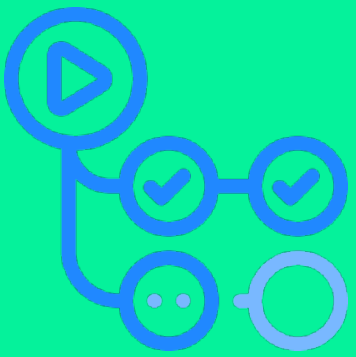
# Taller: Automación de flujo de CI y CD con github actions_



Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

Juan Manuel Ruiz Aranda
jmanuel.ruiz@emerald.dev
@jmruizab

# Objetivo_

Aprender cómo automatizar el flujo de integración y despliegue continuos (CI/CD) de una aplicación utilizando github actions.

Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

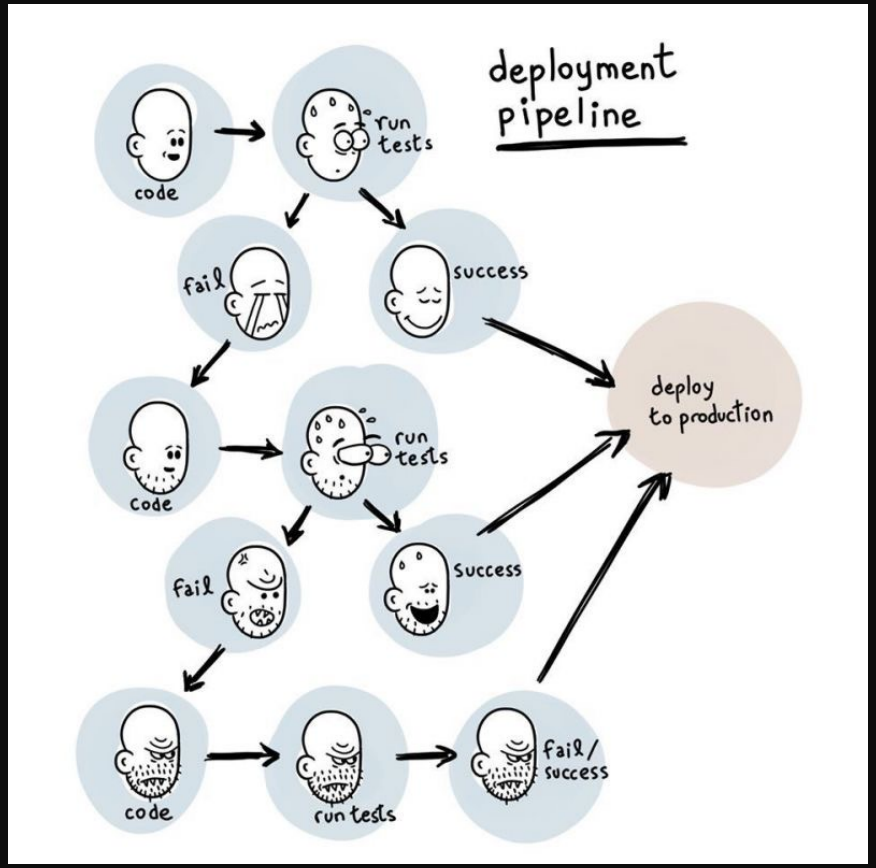
Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

# Agenda_

1. ¿Qué es CI/CD?
2. Github actions
3. Demo
4. Consejos y buenas prácticas

https://github.com/emeraldigital/react-template

# ¿CI/CD?

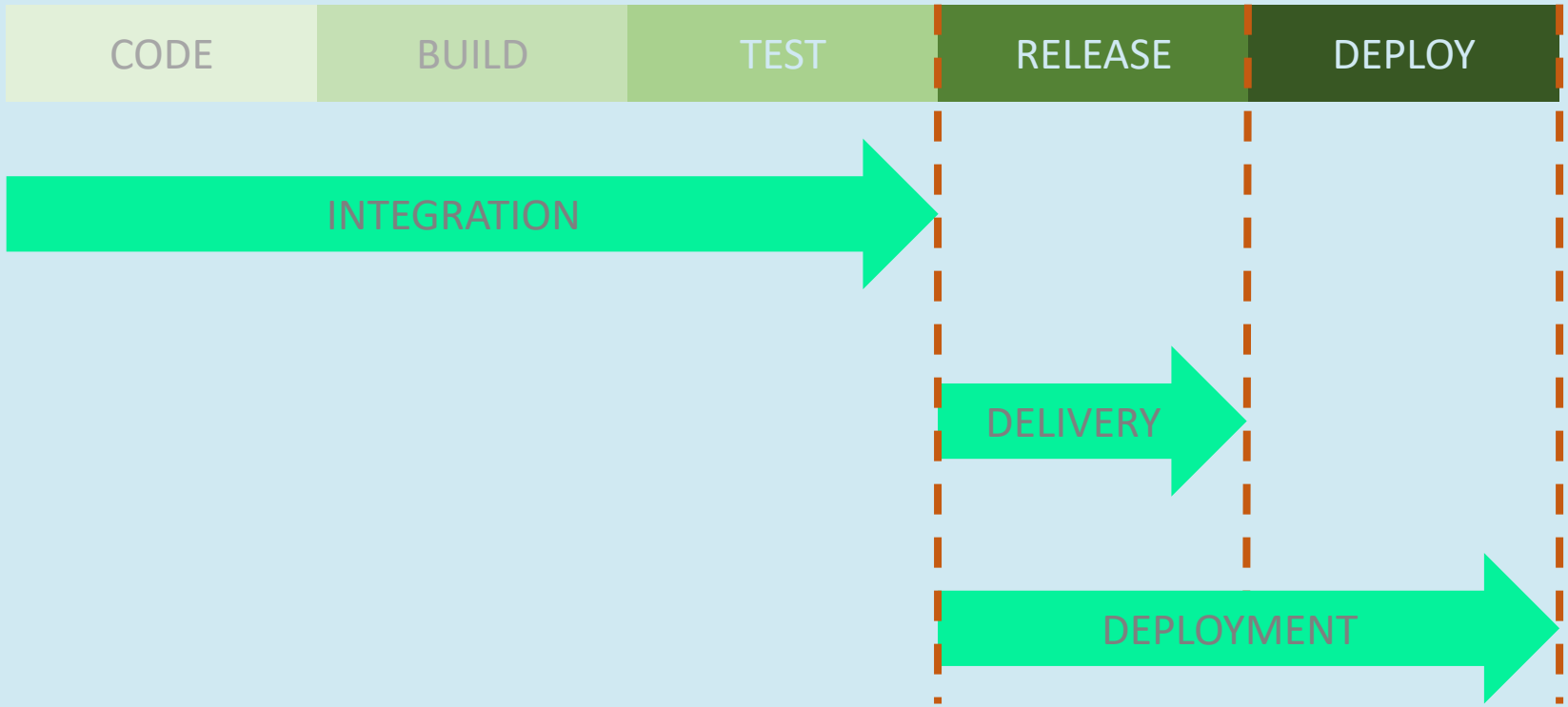



Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

Github Actions

# ¿Qué es CI/CD?_




Emerald

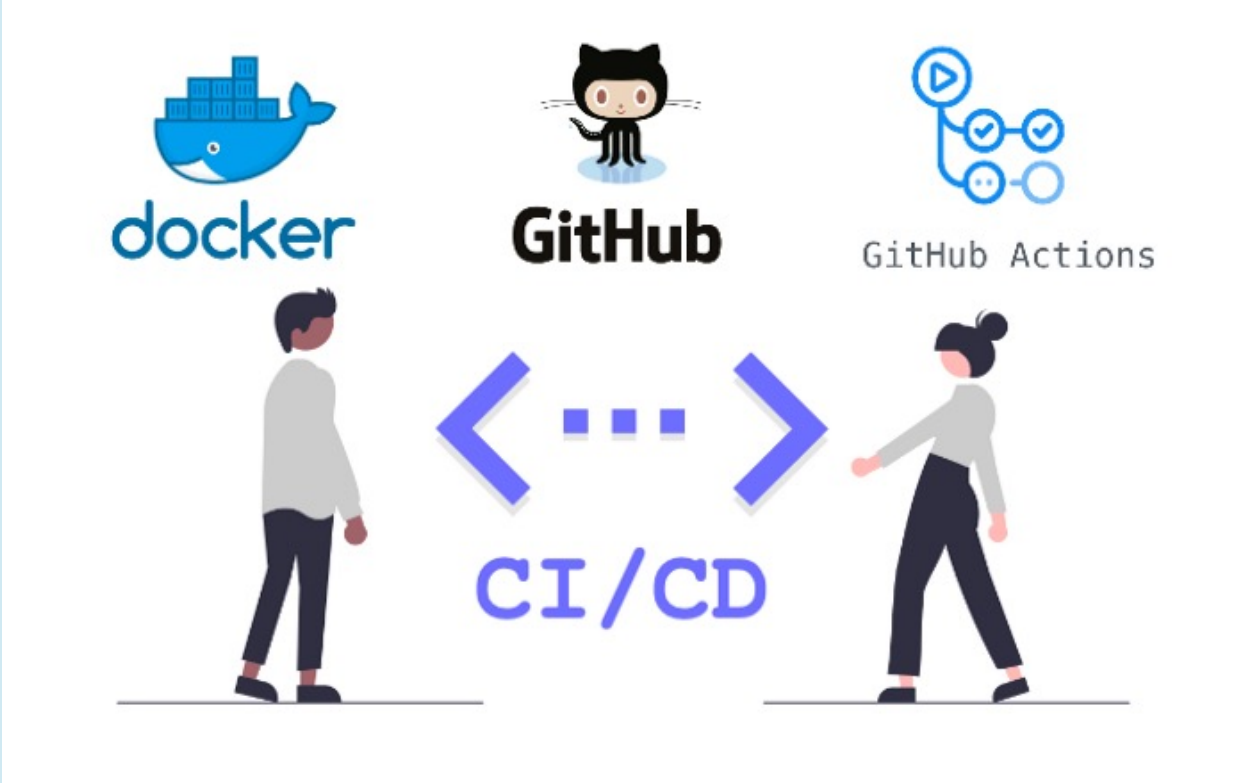
Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

# ¿Cómo?_

Jenkins

Travis CI

GitLab



# ¿Costo?_

| Producto | Almacenamiento | Minutos (por mes) |
| --- | --- | --- |
| GitHub Free | 500 MB | 2,000 |
| GitHub Pro | 1 GB | 3,000 |
| GitHub Free para organizaciones | 500 MB | 2,000 |
| GitHub Team | 2 GB | 3,000 |
| Nube de GitHub Enterprise | 50 GB | 50,000 |

# Github Actions_

Funciona a partir de la definición de workflow's en documentos yaml.

Cada workflow responde a eventos, ejecutando tareas (jobs) compuestas por distintos pasos (steps).

Estos pasos pueden ser acciones o comandos.


Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

# Events_

Los eventos pueden ser:

- Eventos que ocurren en el repositorio de tu flujo de trabajo
- Eventos que ocurren fuera de GitHub (repository_dispatch)
- Programados (Cron Jobs)
- Manual

Eventos (Referencia)

# Jobs_

Los jobs son conjuntos de tareas agrupadas en pasos (steps).

Cada step puede ejecutar comandos ó acciones desarrolladas por terceros.

Cada job se puede ejecutar en distintos runners...


Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

# Runners_

Los runners son contenedores encargados de ejecutar los workflows, podemos utilizar 2 tipos de runners...

```yaml
name: CI-workflow
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - uses: actions/setup-node@v2
    - run: npm install
    - run: npm build
```

- Workflow
- Events
- Runners
- Actions
- Commands

Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

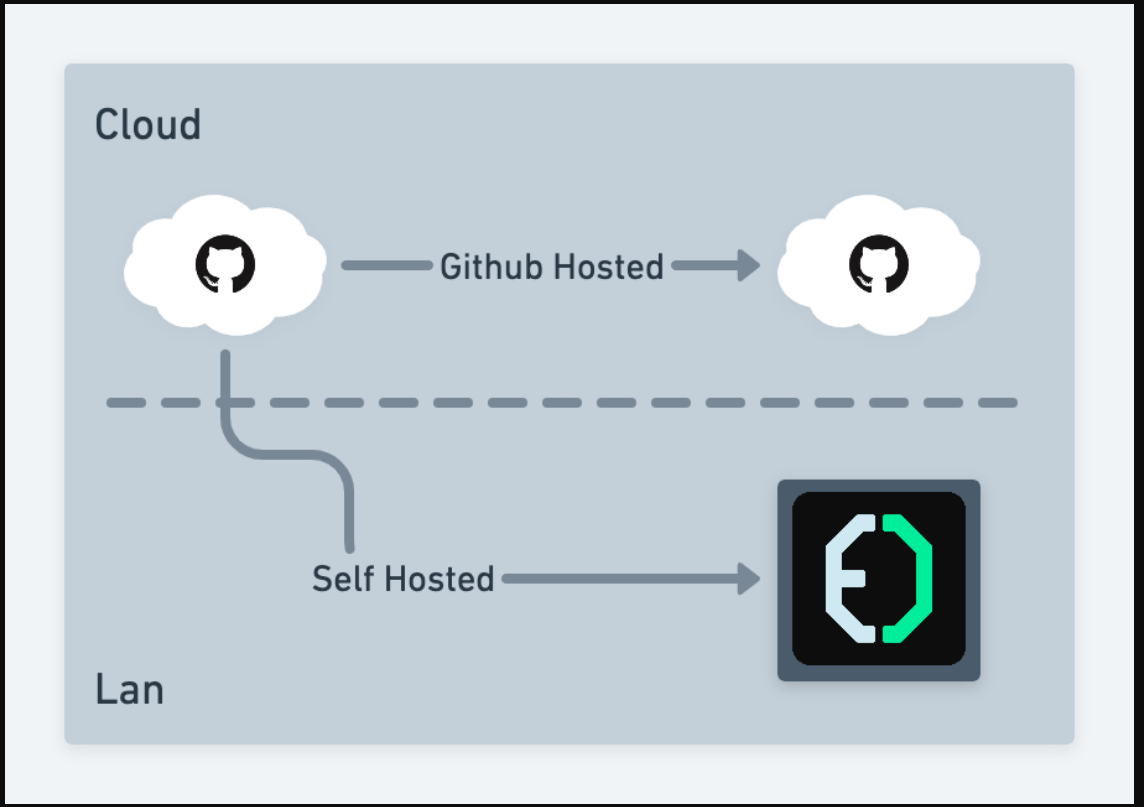
# Github Actions_

### Github Hosted

Son administrados por github y existen tres opciones: Windows, MacOS y Ubuntu y su uso se cobra por minutos.

### Self Hosted

Para soluciones más especificas podemos utilizar recursos propios donde nosotros administremos los recursos de la manera que requiramos.



# Recapitulando_


Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

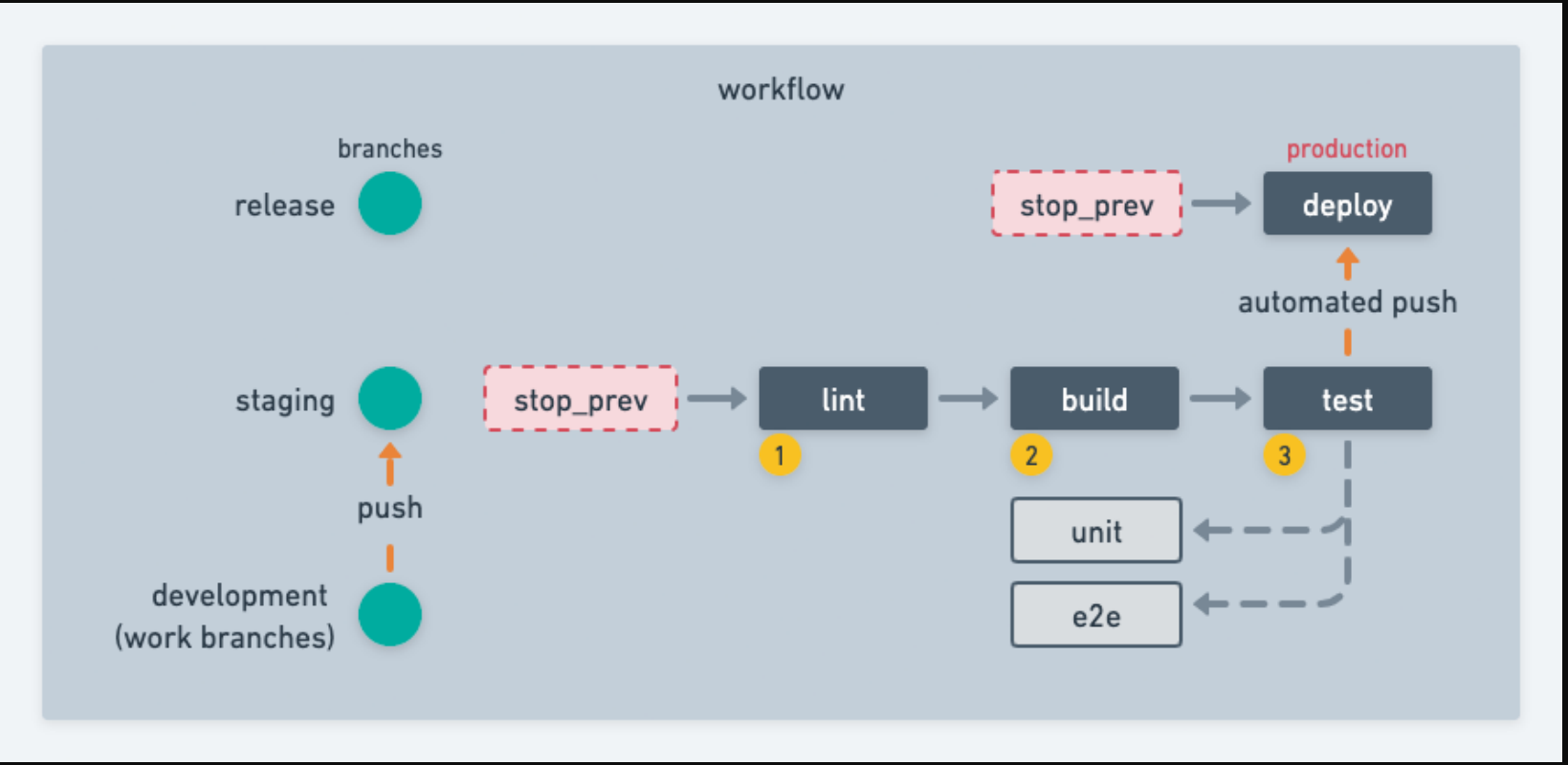
Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

# Demo_

emeraldigital/react-template

Github Actions

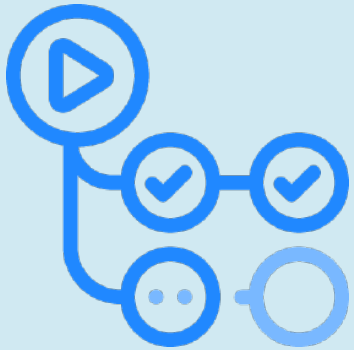
# Demo_




Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

# Caracteristicas Avanzadas_



→ Artefactos (artifacts)
→ Secretos (secrets)
→ Dependencia entre jobs
→ Matrices
→ Almacenar dependencias en caché
→ Contenedores de servicio
→ Uso de tags en workflows
→ Cron Jobs
→ Reutilizar workflows
→ Utilizar ambientes

# Artefactos_

```yaml
jobs:
  build:
    runs-on: ubuntu-20.04
    steps:
      - name: Copy build directory
        uses: actions/upload-artifact@v2
        with:
          name: build
          path: build

  publish:
    needs: [build]
    runs-on: ubuntu-20.04
    steps:
      - name: Copy build directory
        uses: actions/download-artifact@v2
        with:
          name: dist
          path: dist
```

Los artefactos te permiten hacer datos persistentes después de que se complete un job y comparten estos datos con otro job en el mismo flujo de trabajo.

# Secretos_

```yaml
jobs:
  example-job:
    runs-on: ubuntu-latest
    steps:
      - name: Retrieve secret
        env:
          super_secret: ${{ secrets.SUPERSECRET }}
        run: |
          example-command "$super_secret"
```

Se debe evitar el uso de información sensible dentro de nuestros workflows, por lo que podemos utilizar valores secretos, almacenandolos directamente en nuestro repositorio.

[Settings/Actions/Repository's Secret].

# Dependencias_

```yaml
jobs:
  setup:
    runs-on: ubuntu-latest
    steps:
      - run: ./setup_server.sh
  build:
    needs: setup
    runs-on: ubuntu-latest
    steps:
      - run: ./build_server.sh
  test:
    needs: build
    runs-on: ubuntu-latest
    steps:
      - run: ./test_server.sh
```

Por default los jobs se ejecutarán en paralelo.

Sin embargo podemos indicar las dependencias que permitan arrancar un job al termino de la ejecución de otro job.

En caso de que un Job no se complete, no se ejecutarán sus dependencias.

# Uso de caché_

Podemos optimizar nuestros procesos utilizando una caché que estará disponible para todos los flujos de trabajo en el mismo repositorio.

```yaml
jobs:
  example-job:
    steps:
      - name: Cache node modules
        uses: actions/cache@v3
        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 }}-
```

# Servicios_

```yaml
jobs:
  container-job:
    runs-on: ubuntu-latest
    container: node:10.18-jessie
    services:
      postgres:
        image: postgres
    steps:
      - name: Check out repository code
        uses: actions/checkout@v3
      - name: Install dependencies
        run: npm ci
      - name: Connect to PostgreSQL
        run: node client.js
        env:
          POSTGRES_HOST: postgres
          POSTGRES_PORT: 5432
```

Podemos crear contenedores temporales para almacenar servicios.

Los contenedores estarán disponibles para todos los pasos de ese job y se eliminara cuando el job termine.

Servicios (Referencia)

# Matrices, tags y cronjobs_

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        node: [12, 14, 16]
    steps:
      - uses: actions/setup-node@v3
        with:
          node-version: ${{ matrix.node }}
```

Las matrices nos permiten crear y ejecutar multiples jobs a partir de las combinaciones deseadas.

```yaml
jobs:
  example-job:
    runs-on: [self-hosted, linux, x64]
```

Podemos forzar la ejecución de los flujos de trabajo a partir de las etiquetas seleccionadas

```yaml
name: CronJob
on:
  schedule:
    - cron: '0 0 * * *'
```

Podemos ejecutar tareas cada cierto tiempo.

Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

# Consejos y buenas practicas_

→ Debemos evitar utilizar información sensible en nuestros repositorios.
→ En lo posible, debemos optimizar el tiempo de ejecución de las acciones.
→ Procurar especificar las versiones de actions o runners que utilicemos.

Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

# ¿Dudas?_

# Por su atención, Gracias!_

Emerald

Emerald Digital S.C.
Aplicaciones web, ciencia de datos y automatización

CONTACTO
@jmruizab
jmanuel.ruiz@emerald.dev