UNIVERSIDAD DE GUADALAJARA

Centro Universitario de Ciencias Exactas e ingenierías

Computación Tolerante a Fallas

Workflow managers

Nombre: Castillo Mares Gilberto

Código: 213330514

Profesor: Dr. Michel Emanuel López Franco

2024-B

Índice

Contenido

Desarrollo

Prefect es una herramienta de orquestación de flujos de trabajo en Python que facilita la creación, ejecución y monitoreo de flujos de trabajo complejos y dependencias entre tareas. Ofrece una forma de gestionar y programar tareas recurrentes, manejar errores y fallos, y asegurar que los procesos se ejecuten correctamente.

Usaremos esta herramienta de Python para monitorear cada cierto tiempo el estatus de los servicios de PlayStation usando como referencia su pagina web, este ejemplo obtendrá cada 30 segundos una respuesta del servidor para monitorear si está disponible, de no ser este el caso, se mandará una etiqueta en el flujo de la tarea para que esta se marque como fallida y ayude a avisar si están caídos los servicios de PlayStation. Esto lo podremos monitorear utilizando el servicio de Prefect Server con un dashboard en el navegador.

Para comenzar primero necesitamos instalar la herramienta Prefect usando ***pip***. Se recomienda utilizar un entorno virtual en Python:

```
pip install -U prefect
```

Para confirmar que Prefect fue instalado correctamente, usamos este comando:

```
prefect version
```

Después deberíamos poder ver algo similar a esto:

```
Version:             3.0.0
API version:         0.8.4
Python version:      3.12.2
Git commit:          d6bdb075
Built:               Thu, Apr 11, 2024 6:58 PM
OS/Arch:             darwin/arm64
Profile:              local
Server type:         ephemeral
Server:
  Database:          sqlite
  SQLite version:    3.45.2
```

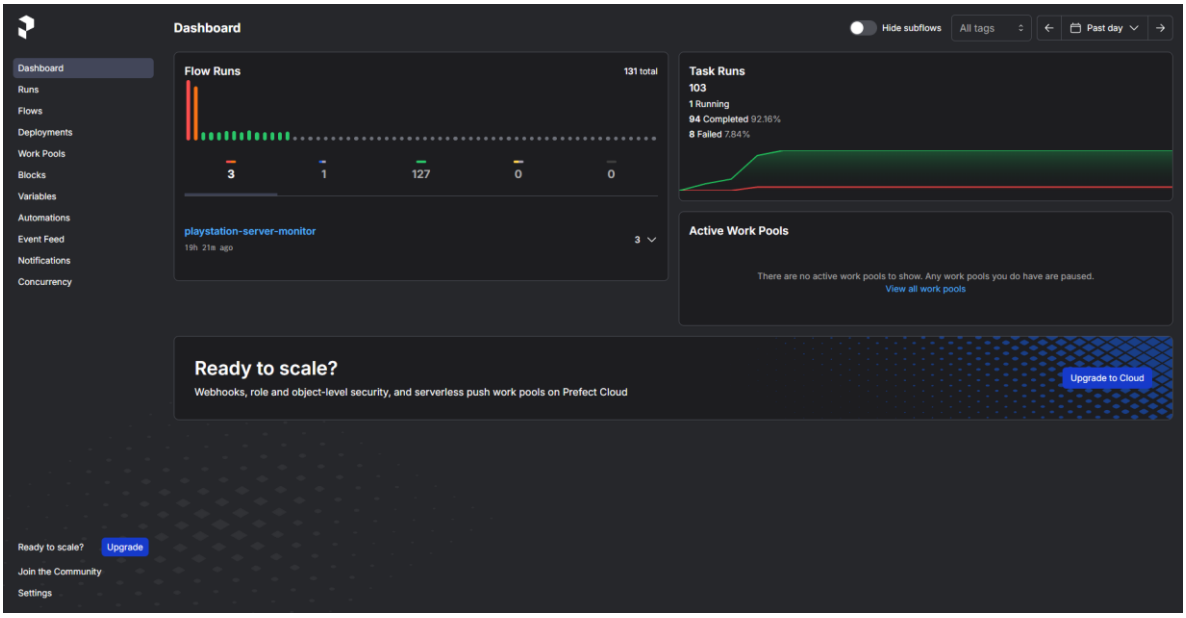
Una vez hecho esto, iniciaremos un servidor API local:

```
prefect server start
```

Nos deberá aparecer este mensaje el cual nos da un enlace directo que abrirá el dashboard del servidor en el navegador:

```
 ___ ___ ___ ___ ___ ___ _____
| _ \ _ \ __| __| __/ __|_   _|
|  _/   / _|| _|| _| (__  | |
|_| |_|_\___|_| |___\___| |_|

Configure Prefect to communicate with the server with:
Configure Prefect to communicate with the server with:
Configure Prefect to communicate with the server with:

    prefect config set PREFECT_API_URL=http://127.0.0.1:4200/api

View the API reference documentation at http://127.0.0.1:4200/docs

Check out the dashboard at http://127.0.0.1:4200
```

Veremos el dashboard en el navegador:



Esta herramienta contiene muchas funciones las cuales no describiré, solamente me enfocaré en lo que utilicé para crear el ejemplo.

Para fines de prueba, usé esta biblioteca en Python para simular respuestas de un servidor no disponible:

```
pip install responses
```

Código script:

Usé este código que da una respuesta dependiendo el estado del servidor, así como utilizar la etiqueta correcta para que la herramienta Prefect indique el estado de la tarea.

```
from prefect import task, flow
from prefect.states import Failed
import requests
import responses
```

- task y flow son decoradores de Prefect que definen tareas y flujos.
- Failed se usa para marcar el estado de una tarea como fallida.

- requests se usa para hacer peticiones HTTP a la página del estado de PlayStation.
- responses se usa para simular respuestas de servidor

```
# Definir la tarea que consulta el estado del servidor
@task
def check_playstation_server_status():
    try:
        # Realizamos una petición GET al endpoint del estado de PlayStation
        response = requests.get("https://status.playstation.com/es-mx/")

        if response.status_code == 200:
            print("El servicio de PlayStation Network esta funcionando
correctamente")
        else:
            print(f"PlayStation server returned status code
{response.status_code}")
            # Marcar el estado como 'Failed' si no se obtiene un código 200
            return Failed(message=f"Servidor PlayStation no disponible.
Status code: {response.status_code}")

    except Exception as e:
        print(f"An error occurred: {e}")
        # Marcar la tarea como fallida si ocurre una excepción
        return Failed(message=f"Error en la tarea: {e}")
```

- Definición de la tarea: Se define una tarea con el decorador @task.
- Petición HTTP: Usa requests.get() para consultar el estado del servidor de PlayStation. El código espera que la respuesta sea un código 200 (éxito).
- Comprobación del estado: Si la respuesta es 200, imprime que el servicio está funcionando correctamente.
- Manejo de errores: Si la respuesta no es 200, se marca como fallida con el estado Failed y un mensaje que incluye el código de estado devuelto.
- Excepciones: Si ocurre alguna excepción durante la petición (problemas de red, por ejemplo), también se marca como Failed y se imprime el error.

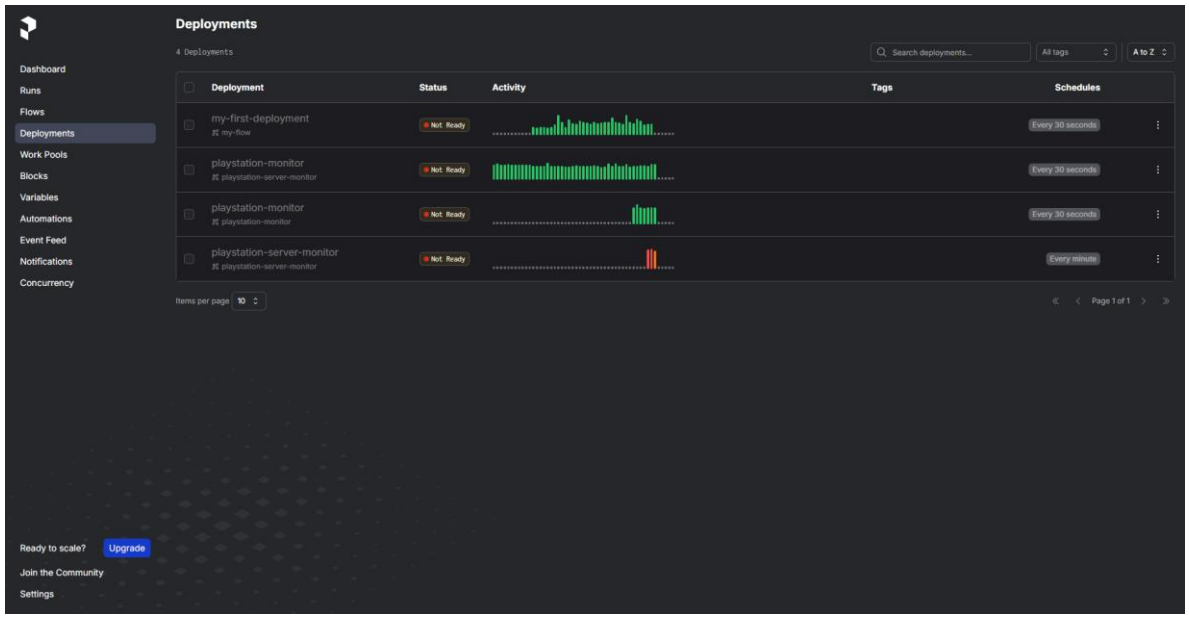
```
# Definir el flujo
@flow
def playstation_monitor():
    check_playstation_server_status()
```

- Definición del flujo: Este es el flujo que encapsula la tarea anterior. Simplemente llama a check_playstation_server_status() para ejecutarla.

```
# Ejecutar el flujo
if __name__ == "__main__":
    playstation_monitor.serve(name="playstation-monitor", interval=30)
```

- Condicional para la ejecución: Este bloque ejecuta el flujo si el script es el programa principal.
- serve(): Ejecuta el flujo en modo "servidor", con un intervalo definido. El parámetro interval=30 indica que el flujo se ejecuta cada 30 segundos.

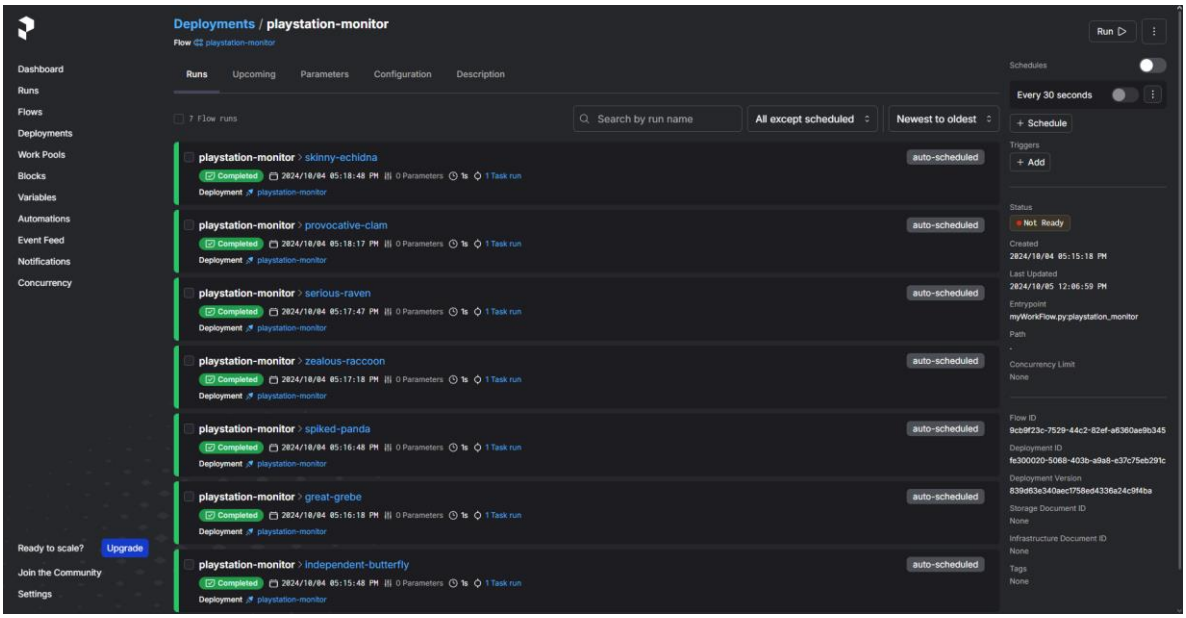
En el dashboard tenemos una opción llamada ***Deployments*** en la cual podemos monitorear todas las implementaciones que hemos hecho utilizando el servidor, en este caso creé una implementación llamada ***playstation-monitor*** la cual deberá aparecer en el dashboard.



Vendría siendo la tercera en la lista.

De momento esta como ***No Ready*** ya que aun no he ejecutado el script en Python para que comience a funcionar.

En esta misma función del dashboard podemos modificar las implementaciones con interfaz, podemos ajustar cada implementación como queramos.



En la imagen podemos observar las pruebas que he estado haciendo, así como la opción de ajustar el intervalo de tiempo, ajustes generales, etc.

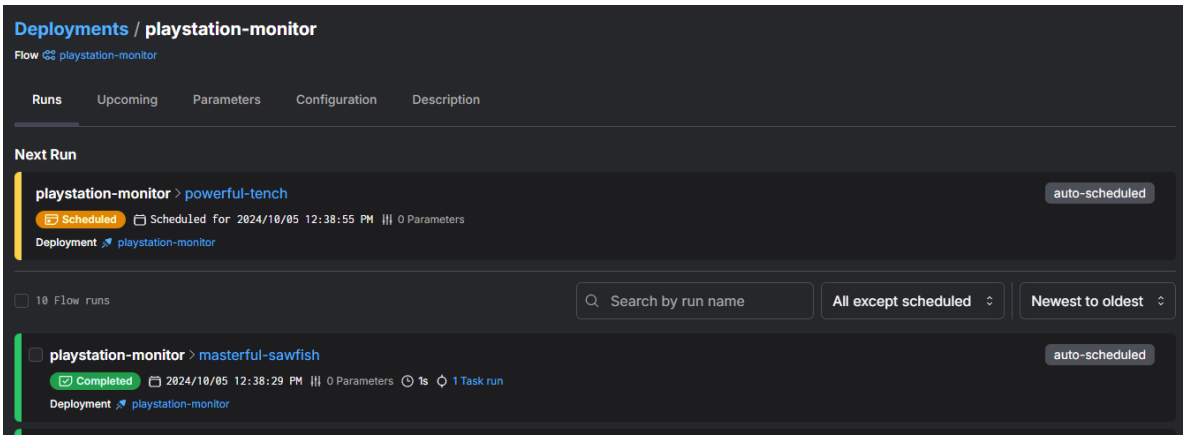
Si es la primera vez que creas una implementación, por lógica, la implementación no se mostrará en el dashboard hasta que ejecutes el script.

Una vez ejecutemos el script, podremos ver mensajes en la consola, así como la implementación en ***Deployments*** en el servidor del navegador.

```
PS F:\2024B\Workflow-Manager> python -u "f:\2024B\Workflow-Manager\myWorkFlow.py"
Your flow 'playstation-monitor' is being served and polling for scheduled runs!

To trigger a run for this flow, use the following command:

        $ prefect deployment run 'playstation-monitor/playstation-monitor'

You can also run your flow via the Prefect UI: http://127.0.0.1:4200/deployments/deployment/fe300020-5068-403b-a9a8-e37c75eb291c

12:37:16.218 | INFO    | prefect.flow_runs.runner - Runner 'playstation-monitor' submitting flow run '11fad070-2bdd-439b-bec8-dfc1d9ef63de'
12:37:16.276 | INFO    | prefect.flow_runs.runner - Opening process...
12:37:16.283 | INFO    | prefect.flow_runs.runner - Completed submission of flow run '11fad070-2bdd-439b-bec8-dfc1d9ef63de'
12:37:21.526 | INFO    | Flow run 'enchanted-peacock' - Downloading flow code from storage at '.'
12:37:26.053 | INFO    | Task run 'check_playstation_server_status-a8e' - Created task run 'check_playstation_server_status-a8e' for task 'check_playstation_server_status'
12:37:26.226 | INFO    | Task run 'check_playstation_server_status-a8e' - Finished in state Completed()
12:37:26.260 | INFO    | Flow run 'enchanted-peacock' - Finished in state Completed()
12:37:26.314 | WARNING | EventsWorker - Still processing items: 3 items remaining...
El servicio de PlayStation Network esta funcionando correctamente
12:37:26.522 | INFO    | prefect.flow_runs.runner - Process for flow run 'enchanted-peacock' exited cleanly.
```

Podemos ver que el flujo se creó y funciona correctamente.

También lo podemos comprobar en el dashboard.



Podemos ver que la tarea ha sido completada correctamente y que esta otra programada para ejecutarse dentro de unos segundos (30).

Así podremos tener control sobre los servicios de PlayStation, ya que cada cierto tiempo el servidor estará monitoreando su estado.

Para el caso contrario, utilizaré la biblioteca ***responses*** para simular una caída del servidor.

Código modificado para simular un servidor caído:

```
from prefect import task, flow
from prefect.states import Failed
import requests
import responses

# Definir la tarea que consulta el estado del servidor
@task
def check_playstation_server_status():
    responses.add(responses.GET, 'https://example.com/api', status=503)
    try:
        # Realizamos una petición GET al endpoint del estado de PlayStation
        response = requests.get("https://example.com/api")

        if response.status_code == 200:
            print("El servicio de PlayStation Network esta funcionando
correctamente")
```

```
        else:
            print(f"PlayStation server returned status code
{response.status_code}")
            # Marcar el estado como 'Failed' si no se obtiene un código 200
            return Failed(message=f"Servidor PlayStation no disponible.
Status code: {response.status_code}")

    except Exception as e:
        print(f"An error occurred: {e}")
        # Marcar la tarea como fallida si ocurre una excepción
        return Failed(message=f"Error en la tarea: {e}")

# Definir el flujo
@flow
def playstation_monitor():
    check_playstation_server_status()

# Ejecutar el flujo
if __name__ == "__main__":
    playstation_monitor.serve(name="playstation-monitor", interval=30)
```

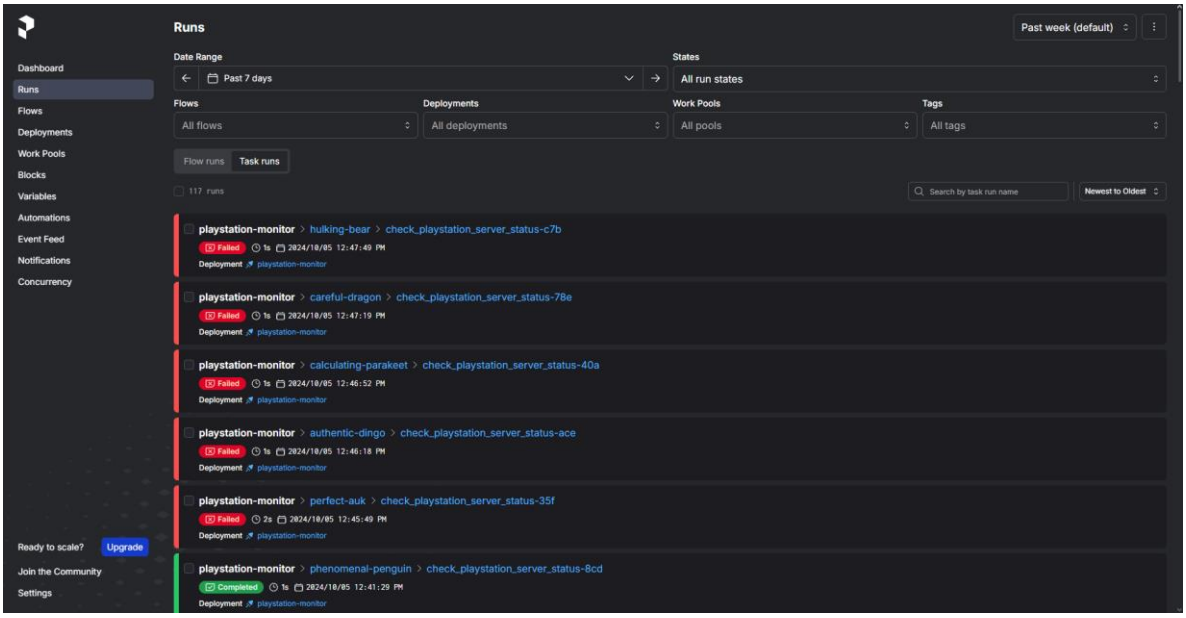
En este ejemplo, cada vez que el código intente hacer una solicitud GET a https://example.com/api, la librería responses simulará que el servidor está caído y devolverá un error 503.

Una vez ejecutado el scrip, podremos ver las pruebas en el dashboard.

```
PS F:\2024B\Workflow-Manager> python -u "f:\2024B\Workflow-Manager\myWorkFlow.py"
Your flow 'playstation-monitor' is being served and polling for scheduled runs!

To trigger a run for this flow, use the following command:

        $ prefect deployment run 'playstation-monitor/playstation-monitor'

You can also run your flow via the Prefect UI: http://127.0.0.1:4200/deployments/deployment/fe300020-5068-403b-a9a8-e37c75eb291c

12:45:43.805 | INFO    | prefect.flow_runs.runner - Runner 'playstation-monitor' submitting flow run 'e5efb56c-072b-4cef-b303-0bc90c6fe85f'
12:45:43.852 | INFO    | prefect.flow_runs.runner - Opening process...
12:45:43.858 | INFO    | prefect.flow_runs.runner - Completed submission of flow run 'e5efb56c-072b-4cef-b303-0bc90c6fe85f'
12:45:46.082 | INFO    | Flow run 'perfect-auk' - Downloading flow code from storage at '.'
12:45:49.139 | INFO    | Task run 'check_playstation_server_status-35f' - Created task run 'check_playstation_server_status-35f' for task 'check_playstation_server_status'
12:45:50.291 | ERROR   | Task run 'check_playstation_server_status-35f' - Finished in state Failed('Servidor PlayStation no disponible. Status code: 404')
12:45:50.327 | INFO    | Flow run 'perfect-auk' - Finished in state Completed()
PlayStation server returned status code 404
12:45:50.502 | INFO    | prefect.flow_runs.runner - Process for flow run 'perfect-auk' exited cleanly.
```

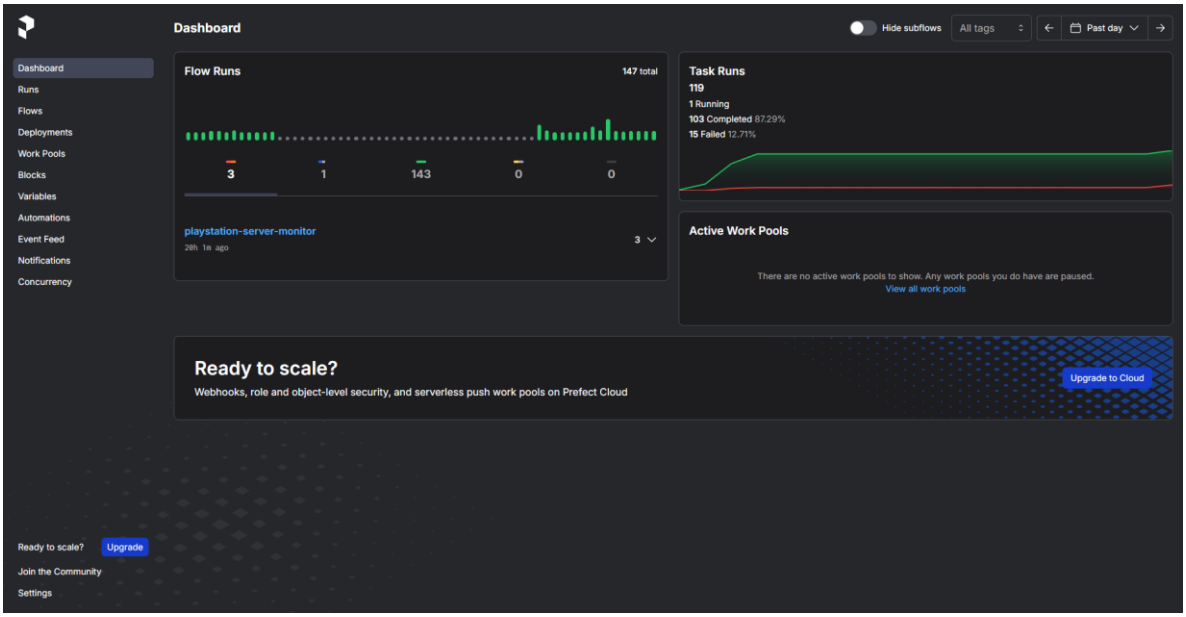
Podemos ver en las impresiones de consola que algo ocurrió con el servidor y mando un error 404. Ahora lo verificamos en el navegador.

Para esto debemos ir a la función ***Runs*** para verificar el estado de los flujos.



Ya ha habido 5 tareas fallidas debido a que el servidor se encuentra caido, aquí las podemos visualizar.

En el panel ***dashboard*** también podemos observar las tareas totales completadas, canceladas y fallidas.



Gracias a esta herramienta podemos crear, ejecutar y monitorear fácilmente tareas.

Bibliografía

Welcome to Prefect - Prefect. (s. f.). Prefect. https://docs.prefect.io/3.0/get-started/index