

# REPORTE Y DOCUMENTACION

## Desafío - Sistema de gestión de tareas

### Participantes:

Francisco Vargas  
Jorge Riquelme  
Diego Jimenez  
Francisco Jara  
Neil Aular  
Fernando Poblete

## DESARROLLO

### 1. Gestión del Repositorio Git

Se inicializa un repositorio local usando el comando 'git init'

```
admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08
$ git init
Initialized empty Git repository in C:/Users/Elitebook/Desktop/PERSONAL/CORFO 2024/DESAFIO08/.git/

admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$
```

Se verifica el estado actual del repositorio git local con el comando 'git status'.

```
admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$ git status
On branch master

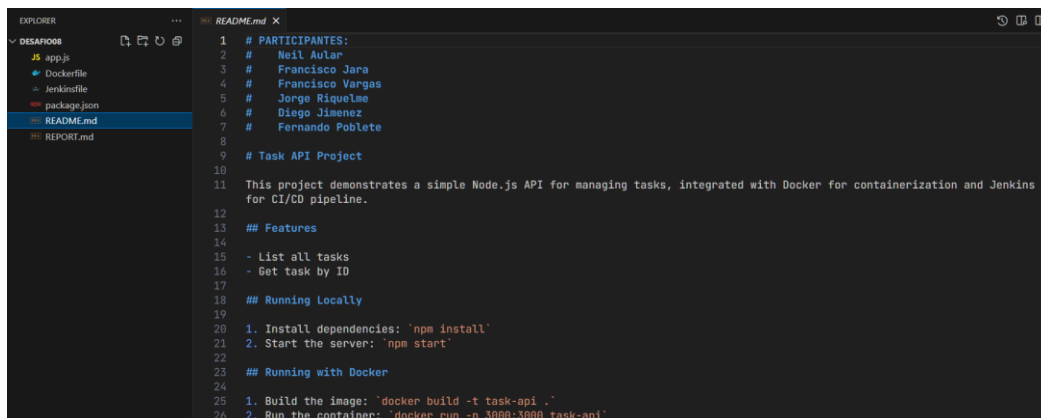
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Dockerfile
    README.md
    REPORT.md
    app.js
    package.json

nothing added to commit but untracked files present (use "git add" to track)

admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$
```

Se modificó el archivo README.md con los integrantes del grupo y se agregaron al stash todos los archivos del proyecto no trackeados con el comando 'git add .'.



The screenshot shows a code editor with the Explorer sidebar on the left displaying a file tree for a project named 'DESAFIO08'. The files listed are 'app.js', 'Dockerfile', 'Jenkinsfile', 'package.json', 'README.md', and 'REPORT.md'. The 'README.md' file is selected and its content is displayed in the main editor area. The content of the README.md file is as follows:

```
1 # PARTICIPANTES:
2 # Neil Aular
3 # Francisco Jara
4 # Francisco Vargas
5 # Jorge Riquelme
6 # Diego Jimenez
7 # Fernando Poblete
8
9 # Task API Project
10
11 This project demonstrates a simple Node.js API for managing tasks, integrated with Docker for containerization and Jenkins
12 for CI/CD pipeline.
13
14 ## Features
15 - List all tasks
16 - Get task by ID
17
18 ## Running Locally
19
20 1. Install dependencies: 'npm install'
21 2. Start the server: 'npm start'
22
23 ## Running with Docker
24
25 1. Build the image: 'docker build -t task-api .'
26 2. Run the container: 'docker run -p 3000:3000 task-api'
```

## REPORTE Y DOCUMENTACION

### Desafío - Sistema de gestión de tareas

```
admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$ git add .
warning: in the working copy of 'Dockerfile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'REPORT.md', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
```

Se finalizó realizando exitosamente un commit con el comando y mensaje ‘git commit -m “readme.md modificado con colaboradores”’

```
admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$ git commit -m "readme.md modificado con colaboradores"
[master (root-commit) 6a8ab0d] readme.md modificado con colaboradores
5 files changed, 111 insertions(+)
create mode 100644 Dockerfile
create mode 100644 README.md
create mode 100644 REPORT.md
create mode 100644 app.js
create mode 100644 package.json
```

Se creó un nuevo repositorio vacío en GitHub.com llamado DESAFIO08 y se agregó como repositorio remoto al git local con el comando ‘git remote add origin https://github.com/byfepo/DESAFIO08.git’

```
admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$ git remote add origin https://github.com/byfepo/DESAFIO08.git

admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
```

Se verifica que el repositorio está configurado exitosamente con el comando ‘git remote -v’

```
admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$ git remote -v
origin https://github.com/byfepo/DESAFIO08.git (fetch)
origin https://github.com/byfepo/DESAFIO08.git (push)

admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$
```

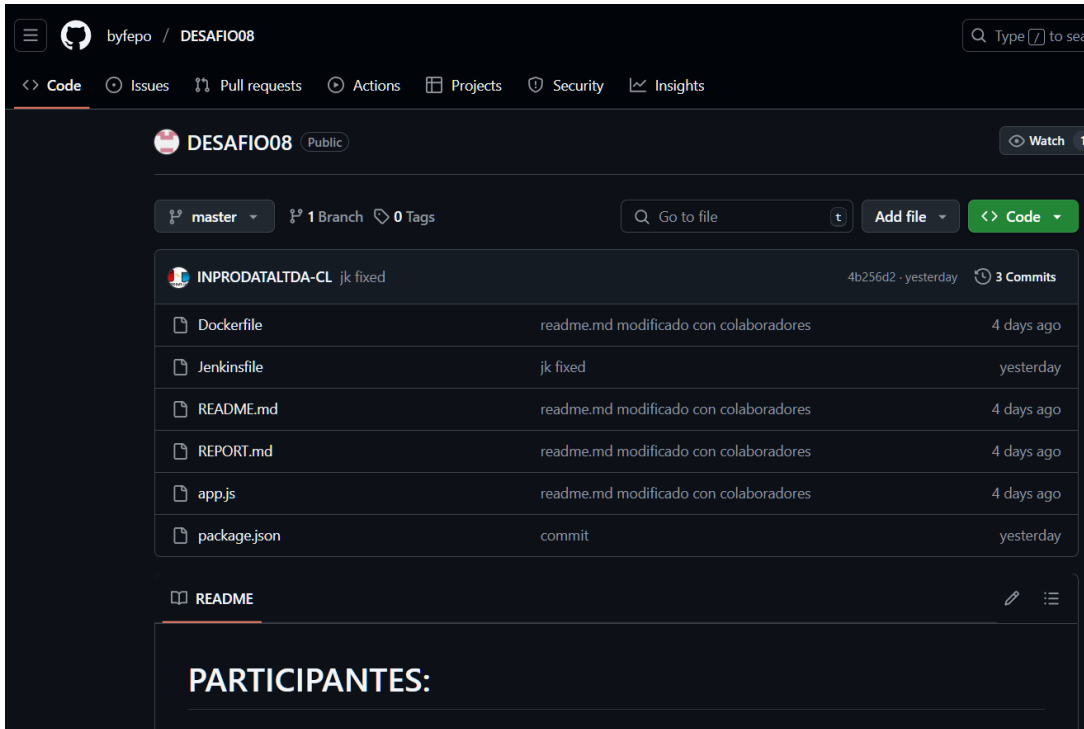
Se realiza el primer push exitoso al repositorio remoto con el comando ‘git push -u origin master’

```
admin@NBINPRODATAi7 MINGW64 ~/Desktop/PERSONAL/CORFO 2024/DESAFIO08 (master)
$ git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.70 KiB | 436.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/byfepo/DESAFIO08.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

# REPORTE Y DOCUMENTACION

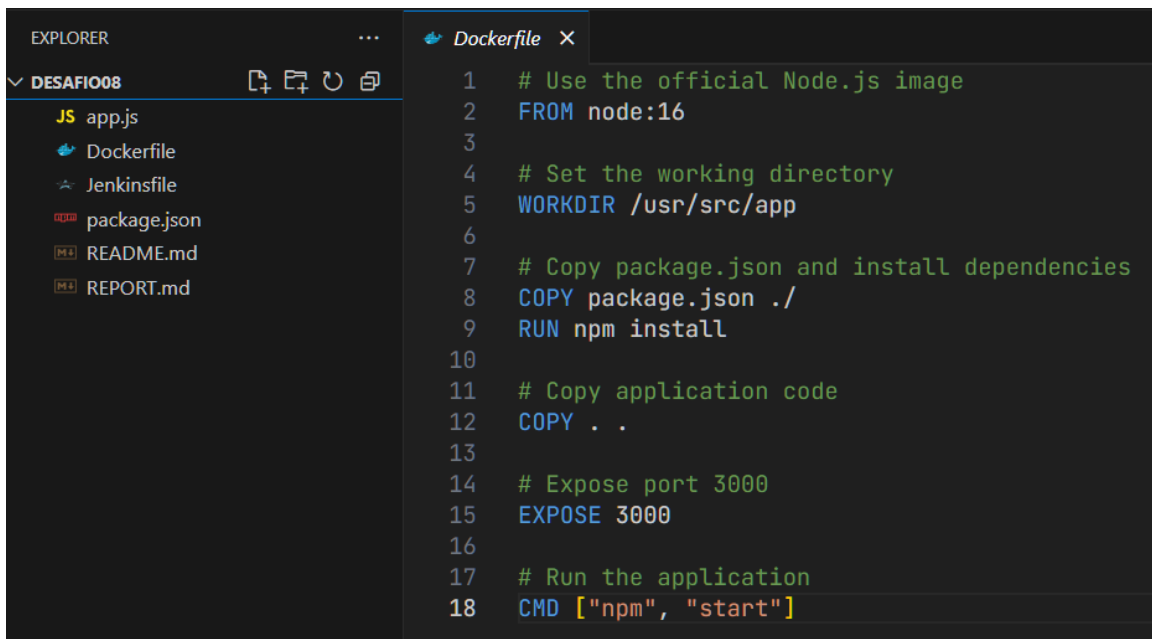
## Desafío - Sistema de gestión de tareas

Se verifica en GitHub.com que el proyecto ha sido subido exitosamente.



## 2. Configuración de la API con Docker

Se presenta un archivo Dockerfile funcional para la creación de una imagen del proyecto NodeJS brindado exponiendo el servicio en el puerto 3000.



# REPORTE Y DOCUMENTACION

## Desafío - Sistema de gestión de tareas

Se verifica que la API del proyecto responde en el puerto 3000 con las rutas /tasks y /tasks/:id

### 3. Configuración Pipeline de Jenkins

Se crea un archivo Jenkinsfile y se configura un Pipeline que permite a Jenkins clonar el repositorio de GitHub, Buildear el proyecto y hacer el Deploy creando una imagen Docker del proyecto en NodeJS.



# REPORTE Y DOCUMENTACION

## Desafío - Sistema de gestión de tareas

```
#9 [5/5] COPY . .
#9 DONE 0.4s

#10 exporting to image
#10 exporting layers
#10 exporting layers 0.4s done
#10 writing image
sha256:a4b7f3470b47e94cb04f5e42953d4b84bda0073e418d291abaf881dfb98e89e6 done
#10 naming to docker.io/library/desafio08grupo02:latest 0.0s done
#10 DONE 0.5s

[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

\*Se anexa un pdf en el archivo zip el Console Output completo de Jenkins con la ejecución completa.

En el equipo personal se crea un contenedor Docker llamado ‘desafio08’, en base a la imagen creada con Jenkins llamada ‘desafio08grupo02:latest’ con el comando ‘docker run -d -p 3000:3000 –name desafio08 desafio08grupo02:latest’

```
PS C:\Users\Elitebook\Desktop\PERSONAL\CORFO 2024\DESAFIO08> docker run -d -p 3000:3000 --name desafio08 desafio08grupo02:latest
0bcb3aef586e253d4211ca1150ddc55fa42557fe5800e33c5895a330a31db3b9
PS C:\Users\Elitebook\Desktop\PERSONAL\CORFO 2024\DESAFIO08>
```

Se verifica en Docker Desktop que el contenedor ‘desafio08’ basado en la imagen ‘desafio08grupo02:latest’ está funcionando correctamente en el equipo personal.

Containers

[Give feedback](#)

View all your running containers and applications.

[Learn more](#)

Container CPU usage

0.00% / 400% (4 CPUs available)

Container memory usage

37.66MB / 7.52GB

Show charts

Search

Only show running containers

<div><input type="checkbox"/></div>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<div><input type="checkbox"/></div>	<div><div></div>desafio08</div>	0bcb3aef586e	<a href="#">desafio08grupo02:latest</a>	<a href="#">3000:3000</a>	0%	38 seconds ago	<div><div></div><div></div><div></div></div>

Se verifica el correcto funcionamiento de la app, probando los endpoints de la API en localhost:3000  
Llamada al endpoint /tasks

```
localhost:3000/tasks
Copiar formato al texto
{"id":1,"name":"Task 1"}, {"id":2,"name":"Task 2"}]
```

Llamada al endpoint /tasks/:id

```
localhost:3000/tasks/2
Copiar formato al texto
{"id":2,"name":"Task 2"}
```