



Diseño e Implementación de Pipeline CI/CD

Continuous Integration (CI)

Estrategia de Branching

Ramas Principales

El trabajo se organiza en dos ramas principales:

- **Main:** Los commits de esta rama están preparados para subir a producción. Es la rama donde se inicia el proyecto y desde donde Jenkins clonará nuestro proyecto. **No se hacen commits aquí.**
- **Develop:** Aquí se encuentra el código que conformará la siguiente versión planificada del proyecto. **No se suelen hacer commits aquí. Solo merges.**

Ramas de Soporte

- **Feature:** se utilizan para desarrollar nuevas características de la aplicación que, una vez terminadas, se incorporan a la rama develop. Llevan por nombre `<feature/nombre-del-feature>`
- **Release:** se utilizan para preparar el siguiente código en producción. En estas ramas se hacen los últimos ajustes y se corrigen los últimos bugs antes de pasar el código a producción incorporándolo a la rama Master. Llevan por nombre `<release/release-version>`

Protección de ramas en github

Para poder proteger el proyecto y aplicar de manera correcta la estrategia de branching, se configuran reglas en Github para las ramas principales, de manera que para cambiarlas se deba hacer un pull request que además lleve una revisión y aprobación. Esto se hace de la siguiente manera:

The screenshot shows the 'Branch protection rule' configuration in GitHub. It has a dark theme. The 'Branch name pattern' is set to 'develop'. Under 'Protect matching branches', two options are checked: 'Require a pull request before merging' and 'Require approvals'. The 'Required number of approvals before merging' is set to 1.

```
branch protection rule
  branch name pattern
    develop
  protect matching branches
    [x] Require a pull request before merging
      When enabled, all commits must be made to a non-protected branch and submitted via a pull request before they can be merged into a branch that matches this rule.
    [x] Require approvals
      When enabled, pull requests targeting a matching branch require a number of approvals and no changes requested before they can be merged.
      Required number of approvals before merging: 1
```

Uso de Github Actions para complementar CI

Para mejorar la protección de las ramas se crean Github Actions para que se ejecuten cuando hay un pull request a las ramas main y develop:


```
25 lines (21 sloc) | 684 Bytes
1 # This workflow will do a clean install of node dependencies, cache/restore them, build the source code and run tests across different versions of node
2 # For more information see: https://help.github.com/actions/language-and-framework-guides/using-nodejs-with-github-actions
3
4 name: Node.js CI
5
6 on:
7   pull_request:
8     branches: [ main, develop ]
9
10 jobs:
11   build_test:
12     name: Build and Test
13     runs-on: ubuntu-latest
14
15     steps:
16     - uses: actions/checkout@v1
17     - name: Use Node.js 16.x
18       uses: actions/setup-node@v1
19       with:
20         node-version: '16.x'
21         cache: 'npm'
22     - run: npm ci
23       working-directory: app
24     - run: npm test
25       working-directory: app
```

☒ **Require status checks to pass before merging**
Choose which **status checks** must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

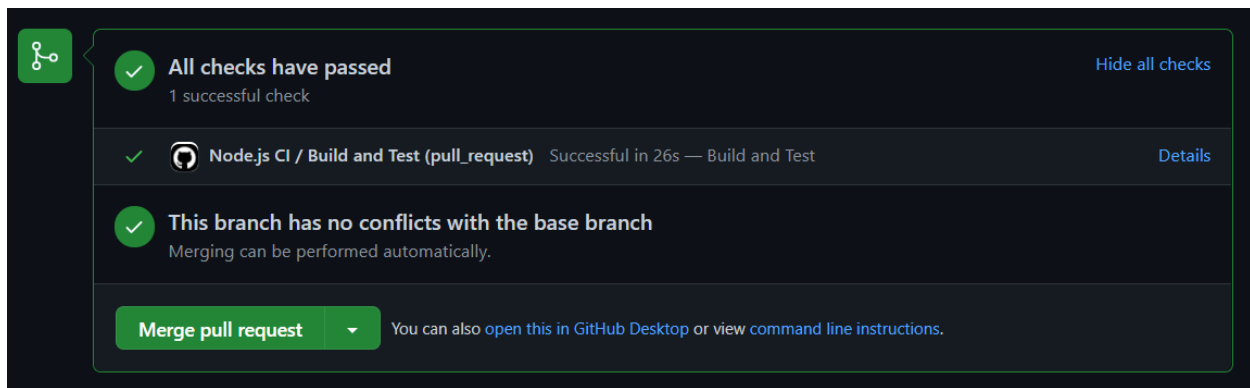
☒ **Require branches to be up to date before merging**
This ensures pull requests targeting a matching branch have been tested with the latest code. This setting will not take effect unless at least one status check is enabled (see below).

Status checks that are required.

build_test

 **GitHub Actions** ▾ ×

Finalmente los pull request a las ramas main y develop se ven de la siguiente manera:



Students App

students-app es una REST API creada con Node.js y Express que expone diferentes endpoints para interactuar con una lista de estudiantes, en donde se encuentra su información básica.

Gestor de paquetes

Dado que es una aplicación hecha con Javascript e interpretada por node se utiliza el gestor de paquetes npm.

Frameworks de Testing

Se utilizan 2 frameworks para hacer testing de la aplicación:

- Jest: Es un framework simple y fácil de utilizar que permite hacer pruebas en javascript. Se decidió utilizar por la facilidad de aprendizaje y su buena documentación.
- Supertest: Es un pequeño módulo que proporciona una abstracción de alto nivel para probar HTTP, al mismo tiempo que le permite acceder a la API de nivel inferior proporcionada por superagent. Este módulo se usa para poder probar las respuestas de los endpoints de la API.

Configuración del Repo de Github y Jenkins

En Jenkins se crea el nuevo pipeline:

A screenshot of the Jenkins 'Enter an item name' form. The form has a text input field containing 'students-app-pipeline' and a 'Required field' label. Below the input field, there are two options: 'Crear un proyecto de estilo libre' (Create a free-style project) and 'Pipeline'. The 'Pipeline' option is selected. The description for 'Pipeline' reads: 'Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.'

Se configura el pipeline para que tome el Jenkinsfile proveniente de el repositorio y solo se construye la rama main:

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/snaranjop1/students-app

Credentials

- none - Add

Avanzado...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/main

Guardar Apply Add Branch

Para que el pipeline se ejecute cuando hay un push en la rama main, se debe crear un webhook en github donde agregamos la url de la instancia jenkins y el token que se puede encontrar en la parte de credenciales de la configuración de jenkins:

Options

Manage access

Security & analysis

Branches

Webhooks

Notifications

Integrations

Deploy keys

Actions

Environments

Secrets

Pages

Moderation settings

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

https://example.com/postreceive

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ **Active**
We will deliver event details when this hook is triggered.

Add webhook

Ademas en jenkins se configura para que escuche el webhook de github:

Build Triggers

☐ Construir tras otros proyectos

☐ Ejecutar periódicamente

☒ GitHub hook trigger for GITScm polling

☐ Consultar repositorio (SCM)

☐ Desactivar la ejecución

☐ Periodo de espera

☐ Lanzar ejecuciones remotas (ejem: desde 'scripts')

?

?

?

?

?

?

?

Continuous Delivery (CD)

Por facilidad de diseño e implementacion se decide que las imagenes de docker donde se containeriza la aplicacion *students-app* se subiran al registro publico de docker (Docker hub).

Versionamiento



Se decide utilizar el Versionamiento Semántico el cual es un estándar a la hora de definir la versión de una API pública.

Patch: Es el tercer número (derecha a izquierda), se refiere principalmente a corrección de errores compatibles con versiones anteriores.

Minor: Es el número del centro y se modifica cuando aparecen corrección de errores y/o mejoras en funcionalidades o nuevas funcionalidades que no son cruciales para el proyecto. Si se aumenta la versión “*minor*” se reiniciar la versión “*patch*”.

Major: Indica que el API ha sufrido un gran cambio y que además, ese cambio puede ocasionar errores para la gente que la utiliza. El cambio puede incluir corrección de errores y/o mejoras en funcionalidades o nuevas características/funcionalidades. Cuando se incrementa la versión “*major*”, entonces “*minor*” y “*patch*” se reinician.

Delivery

Se suben las imagenes a Docker Hub y la version en el tag esta determinada por el tag que tiene el HEAD de la rama Main de esta manera se mantiene una coherencia entre las versiones que se encuentran tageadas en el repositorio y las que se encuentran publicadas en el registro de docker.

Sort by

Newest

Filter Tags

TAG

v1.2.0

Last pushed a few seconds ago by snaranjomaster

DIGEST

5edab8075961

OS/ARCH

linux/amd64

LAST PULL

a few seconds ago

COMPRESSED SIZE

52.81 MB

docker pull snaranjomaster/student-...

TAG

v1.1.0

Last pushed 5 days ago by snaranjomaster

DIGEST

89e58e1b01c5

OS/ARCH

linux/amd64

LAST PULL

an hour ago

COMPRESSED SIZE

52.87 MB

docker pull snaranjomaster/student-...

TAG

v1.0.0

Last pushed 5 days ago by snaranjomaster

DIGEST

52d8c776be8f

OS/ARCH

linux/amd64

LAST PULL

5 days ago

COMPRESSED SIZE

52.87 MB

docker pull snaranjomaster/student-...

Releases	Tags
Tags	
v1.2.0	
1 hour ago b617414 zip tar.gz Notes	
v1.1.0	
5 days ago 1b2c52c zip tar.gz Notes	
v1.0.0	
5 days ago 1780e38 zip tar.gz Notes	

Continuous Deployment (CD)

Para el despliegue se decide utilizar un Servicio tipo LoadBalancer para exponer el servicio en una ip pública. buscando alta disponibilidad se despliega la aplicacion en un Deployment de k8s con 3 replicas y con pods restringidos en terminos de cpu y memoria.

Resultado Final

Tras diseñar el pipeline, configurar jenkins y escribir el Jenkinsfile, estos son los pasos definidos para el pipeline completo:

```
pipeline {
  agent any
  stages {
    stage('Build API') {
      steps {
        dir('app') {
          echo 'npm install'
```

```

    }
  }
}
stage('Test API') {
  steps {
    dir('app') {
      echo 'npm test'
    }
  }
}
stage('Build docker image') {
  steps {
    echo 'docker build -t snaranjomaster/student-app:latest -t snaranjomaster/student-app:${git describe --tags --abbrev=0} .'
  }
}
stage('Push docker image to registry') {
  steps {
    echo 'docker push snaranjomaster/student-app -a'
  }
}
stage('Deployment') {
  steps {
    echo 'kubectl apply -f students-app.yaml'
  }
}
}
}
}

```

Usando el plugin de Blue Ocean en Jenkins:

