

Proyecto de software basado en herramientas de integración continua

Integración continua

Entrega Semana 7

Integrantes:

Lisbeth Alvarez Zuluaga

Andres Felipe Triana Gallego

Politécnico Grancolombiano

Ingeniería de Software

2023

Contenido:.....	2
-----------------	---

### **Entrega semana 3**

Proyecto a realizar y lenguaje .....	3
--------------------------------------	---

Justificación .....	4
---------------------	---

Alcances.....	5
---------------	---

Enlace repositorio y Contexto del código .....	6
--	---

### **Entrega semana 5**

Implementación Jenkins .....	8
------------------------------	---

### **Entrega semana 7**

Travis CI.....	16
----------------	----

CodeShip.....	21
---------------	----

Historial de cambios .....	32
----------------------------	----

## **Proyecto a realizar y lenguaje**

Realizaremos un ecommerce que nos permita poner en práctica todo lo aprendido, además de generar un entregable de alta confiabilidad, alta y rápida respuesta, fácil escalabilidad e integración. Para esto utilizaremos como base:

**Front-end:** JavaScript es esencial para el desarrollo del front-end de un sitio web de comercio electrónico. Aquí podemos utilizar bibliotecas y frameworks como React, Angular o Vue.js para crear interfaces de usuario interactivas y atractivas. Estas tecnologías nos permiten construir páginas de producto, carritos de compra, formularios de registro y otros componentes front-end.

**Back-end:** Para el desarrollo del back-end, Node.js nos parece una excelente opción. Node.js nos permite utilizar JavaScript en el lado del servidor. Puedes construir una API RESTful o GraphQL para gestionar la lógica empresarial, manejar la autenticación de usuarios, procesar pedidos y gestionar la base de datos.

**Base de datos:** JavaScript no es un lenguaje de base de datos, por lo que necesitaremos una base de datos para almacenar la información de productos, pedidos, usuarios, etc. Usaremos bases de datos relacionales MySQL.

## **Justificación**

Crear un ecommerce con las características mencionadas en la respuesta anterior que permita emplear herramientas para la integración continua, utilizando JavaScript tanto en el front-end como en el back-end, nos da ventajas y aspectos positivos. Aquí hay algunos de ellos:

### **Ventajas:**

**Versatilidad tecnológica:** El uso de JavaScript en el front-end y Node.js en el back-end nos brinda una plataforma versátil y unificada para el desarrollo.

**Interactividad y experiencia de usuario mejorada:** JavaScript nos permite crear interfaces de usuario altamente interactivas y atractivas. Los frameworks como React, Angular y Vue.js son ideales para proporcionar a los usuarios una experiencia de compra más agradable y personalizada.

**Eficiencia en el desarrollo:** Al utilizar JavaScript en todo el ciclo de desarrollo, nos permite optimizar la eficiencia y la velocidad de desarrollo.

**Comunidad y recursos abundantes:** JavaScript tiene una comunidad de desarrollo activa y una gran cantidad de recursos, tutoriales y bibliotecas disponibles para facilitar el desarrollo y resolver problemas comunes.

Escalabilidad y rendimiento: Node.js es conocido por su capacidad de manejar un alto volumen de conexiones simultáneas, lo que lo hace adecuado para sitios web de comercio electrónico con un tráfico creciente.

Seguridad: Con un enfoque adecuado en la seguridad, podemos garantizar que las transacciones en línea y los datos de los usuarios estén protegidos.

Flexibilidad en la elección de bases de datos: JavaScript nos permite elegir entre una variedad de bases de datos, ya sea una base de datos relacional o NoSQL, según nuestras necesidades y preferencias.

### **Alcances:**

Gestión de productos: Poder administrar eficientemente una amplia variedad de productos, descripciones, precios y categorías en nuestro ecommerce.

Procesamiento de pedidos: Lograr automatizar el proceso de procesamiento de pedidos, desde la selección de productos hasta la confirmación y envío.

Gestión de usuarios y autenticación: Poder implementar un sistema de autenticación seguro para que los usuarios puedan crear cuentas, iniciar sesión y realizar un seguimiento de sus pedidos.

Pasarelas de pago: Integrar pasarelas de pago seguras y confiables para procesar transacciones en línea.

Seguimiento de inventario: Realizar seguimientos del inventario en tiempo real y gestiona el stock de productos.

Análisis y métricas: Implementar herramientas de análisis para rastrear el rendimiento de la tienda, las conversiones y el comportamiento de los usuarios.

Personalización y recomendaciones: Poder personalizar la experiencia de compra de los usuarios y ofrecer recomendaciones de productos relevantes.

- **Enlace del repositorio:** <https://github.com/lisalvarez18/proyecto-ecommerce>
- Estructura básica del proyecto node js la cual fue creada usando el administrador de paquetes NPM.

```

1 {
2   "name": "proyecto-ecommerce",
3   "version": "1.0.0",
4   "description": "💎💎#\u0000 \u0000p\u0000r\u0000o\u0000y\u0000e\u0000c\u0000t\u0000o\u0000-\u0000e\u0000c\u0000o\u0000m\u0000m\u0000",
5   "main": "app.js",
6   "scripts": {
7     "test": "echo \"Error: no test specified\" && exit 1"
8   },
9   "repository": {
10    "type": "git",
11    "url": "git+https://github.com/lisalvarez18/proyecto-ecommerce.git"
12  },
13  "author": "",
14  "license": "ISC",
15  "bugs": {
16    "url": "https://github.com/lisalvarez18/proyecto-ecommerce/issues"
17  },
18  "homepage": "https://github.com/lisalvarez18/proyecto-ecommerce#readme"
19 }
20

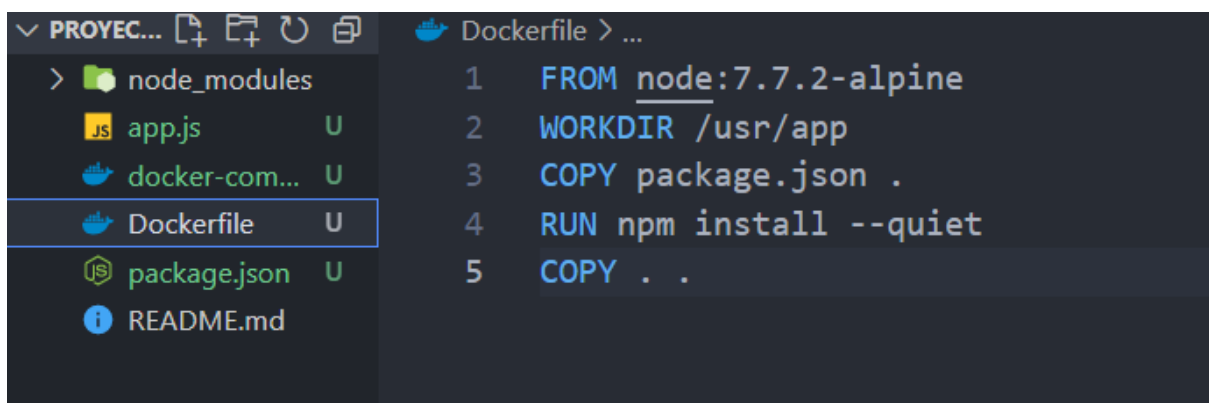
```

- Hola mundo en node js usando un servidor http.



```
1
2 var http = require('http');
3 var server = http.createServer();
4
5 function mensaje(petic, resp) {
6   resp.writeHead(200, {'content-type': 'text/plain'});
7   resp.write('Hola Mundo');
8   resp.end();
9 }
10 server.on('request', mensaje);
11
12 server.listen(3000, function () {
13   console.log('La Aplicación está corriendo en el puerto 3000');
14 });
```

- Este código se usa para crear una imagen de la aplicación.



```
1 FROM node:7.7.2-alpine
2 WORKDIR /usr/app
3 COPY package.json .
4 RUN npm install --quiet
5 COPY . .
```

- Este código se usa para levantar dos contenedores, uno basado en la imagen de nuestra aplicación y el otro basado en la imagen de la base de datos de postgres.

```
1  version: '2'
2  services:
3    web:
4      build: .
5      command: node app.js
6      volumes:
7        - ./usr/app/
8        - /usr/app/node_modules
9      ports:
10       - "3000:3000"
11      depends_on:
12        - postgres
13    postgres:
14      image: postgres:9.6.2-alpine
15      environment:
16        POSTGRES_USER: admin
17        POSTGRES_DB: admin
```

## Implementación Jenkins

- En este archivo docker file creamos una imagen propia de Jenkins la cual va a tener instalado Docker.

```
jenkins > Dockerfile > ...
1  FROM jenkins/jenkins:lts
2  USER root
3  RUN apt-get update
4  RUN curl -sSL https://get.docker.com/ | sh
```

El siguiente comando se usa para construir nuestra imagen de Jenkins.

```
docker build -t lisbeth/jenkins .
```

Este comando es para correr nuestra imagen de jenkins con los respectivos puertos y volúmenes.



```
docker run -p 8080:8080 -p 50000:50000 -v
```

```
<Jenkins_home>:/var/jenkins_home -v
```

```
/var/run/docker.sock:/var/run/docker.sock lisbeth/Jenkins
```

- Luego de que corremos nuestro contenedor de Jenkins se nos proporcionará a través de los logs un password para iniciar la configuración de Jenkins.

```
2023-11-16 21:32:46.013+0000 [id=31] INFO jenkins.install.SetupWizard#init:
*****
*****
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

de7e99dd77aa49a28839c72bdc429351

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
2023-11-16 21:33:33.841+0000 [id=47] INFO h.m.DownloadService$Downloadable#load: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
2023-11-16 21:33:33.842+0000 [id=47] INFO hudson.util.Retrier#start: Performed the action check updates server successfully at the attempt #1
2023-11-16 21:33:34.280+0000 [id=31] INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization
```

- Al configurar Jenkins nos pedirá algunos datos (nombre, usuario, contraseña y email) y adicional se sugerirán algunos plugins, en nuestro caso instalamos los sugeridos más node js.
- En Jenkins procedemos a crear nuestra tarea de pipeline.

## Enter an item name

» Required field



### Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.



### Pipeline

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.



### Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

New Item de tipo Pipeline, OK.

- Comenzamos la configuración e ingresamos una breve descripción.

## General

Enabled

### Descripción

Pipeline para construir testear y desplegar nuestro proyecto e-commerce usando npm

Plain text [Visualizar](#)

- Configuramos nuestro pipeline para que funcione desde un script de SCM (supply chain management), y en esta ventana configuramos la url y las credenciales de nuestro repositorio, para que de esta forma jenkins tenga acceso al código.

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/lisalvarez18/proyecto-ecommerce

Credentials ?

lisalvarez18/\*\*\*\*\*

+ Add

Avanzado

Guardar Apply

- Configuramos la rama de la cual Jenkins va a obtener el código.

Branches to build ?

Branch Specifier (blank for 'any') ?

\*/main

- En este campo de texto ingresamos el path de nuestro Jenkins file, el cual contiene la configuración de nuestro pipeline.

Script Path ?

Jenkinsfile

- Este es el Código de nuestro “Hola mundo”, el cual fue modificado para facilitar los test.

```
js app.js > ...
1  const express = require("express");
2  const app = express();
3  const port = process.env.PORT || 3000;
4
5  app.get(["/", "/index.html"], (req, res) => {
6    res.send("Hola mundo");
7  });
8
9  if (process.env.NODE_ENV !== "test") {
10    app.listen(port, () => {
11      console.log(`La aplicacion esta corriendo en el puerto ${port}`);
12    });
13  }
14
15  module.exports = app;
```

- Estas corresponden a nuestras pruebas unitarias.

```
js test.js > ...
1  const server = require("./app");
2  const supertest = require("supertest");
3  const { default: expect } = require("expect");
4
5  describe("Homepage", () => {
6    const checkHome = (supertest_response) => {
7      expect(supertest_response.status).toEqual(200);
8      expect(supertest_response.type).toEqual('text/html');
9      expect(supertest_response.text).toEqual("Hola mundo");
10   }
11   it("GET / return Hola mundo", async () => {
12     const res = await supertest(server).get("/");
13     checkHome(res);
14   });
15   it("GET /index.html return Hola mundo", async () => {
16     const res = await supertest(server).get("/index.html");
17     checkHome(res);
18   });
19 });
```

- La siguiente corresponde a nuestro Jenkins file que contiene nuestra configuración para nuestro pipeline en Jenkins. En este mismo archivo podemos ver las diferentes etapas de nuestro pipeline.

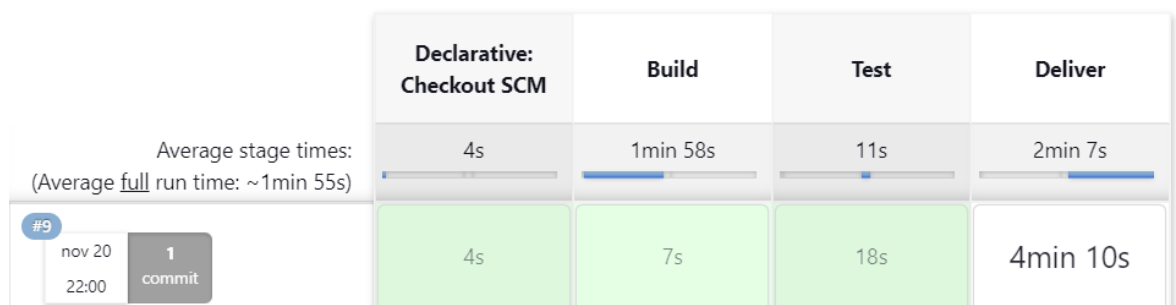
```

Jenkinsfile
1 pipeline {
2     agent {
3         docker {
4             image 'node:20.9.0-alpine3.18'
5             args '-p 3000:3000'
6         }
7     }
8     environment {
9         CI = 'true'
10    }
11    stages {
12        stage('Build') {
13            steps {
14                sh 'npm install'
15            }
16        }
17        stage('Test') {
18            steps {
19                sh 'npm test'
20            }
21        }
22        stage('Deliver') {
23            steps {
24                sh 'node app.js'
25                input message: 'Finished using the web site? (Click "Proceed" to continue)'
26                sh 'kill -INT 888'
27            }
28        }
29    }
30 }

```

Aquí Podemos ver una captura de cada una de las etapas del pipeline en la consola de Jenkins.

## Stage View



- En estos logs que arroja Jenkins cuando ejecutamos nuestro pipeline, podemos apreciar como Jenkins descarga nuestro código desde nuestro repositorio.

```

Started by user Lisbeth Alvarez
Obtained Jenkinsfile from git https://github.com/lisalvarez18/proyecto-ecommerce
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/e-commerce-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
using credential 9bf8bebb-8b77-4817-91e5-8073a45717a0
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/e-commerce-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/lisalvarez18/proyecto-ecommerce # timeout=10
Fetching upstream changes from https://github.com/lisalvarez18/proyecto-ecommerce
> git --version # timeout=10
> git --version # 'git version 2.39.2'
using GIT_ASKPASS to set credentials
> git fetch --tags --force --progress -- https://github.com/lisalvarez18/proyecto-ecommerce +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 28475dcac0a6133978258f8b54591f8e39d8133c (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 28475dcac0a6133978258f8b54591f8e39d8133c # timeout=10
Commit message: "Borrando npm build"
> git rev-list --no-walk e466a87ebda9378d587b0f708d19a3dcb1175402 # timeout=10

```

En las siguientes capturas Podemos apreciar cada una de las etapas de nuestro pipeline.

- **Build:** En esta etapa ejecutamos el comando npm install , el cual descarga todas las dependencias de nuestro proyecto.

```

[Pipeline] { (Build)
[Pipeline] sh
+ npm install

up to date, audited 374 packages in 2s

44 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

```

- **Test:** En esta etapa se ejecutan las pruebas unitarias.

```
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] sh
+ npm test

> proyecto-ecommerce@1.0.0 test
> cross-env CI=true NODE_ENV=test jest --testTimeout=10000 --detectOpenHandles --forceExit

PASS ./test.js (6.886 s)
  Homepage
    ✓ GET / return Hola mundo (152 ms)
    ✓ GET /index.html return Hola mundo (17 ms)

Test Suites: 1 passed, 1 total
Tests:      2 passed, 2 total
Snapshots:  0 total
Time:       7.291 s
Ran all test suites.
```

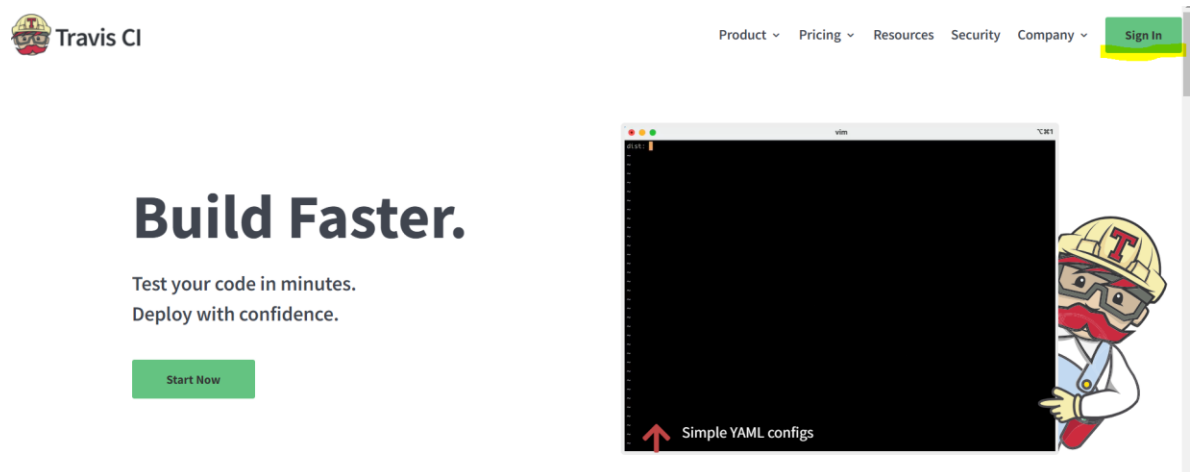
- **Deliver:** En esta etapa se ejecuta la aplicación y se detiene intencionalmente después de unos minutos.

```
[Pipeline] stage
[Pipeline] { (Deliver)
[Pipeline] sh
+ node app.js
La aplicacion esta corriendo en el puerto 3000
Sending interrupt signal to process
Aborted by Lisbeth Alvarez
```

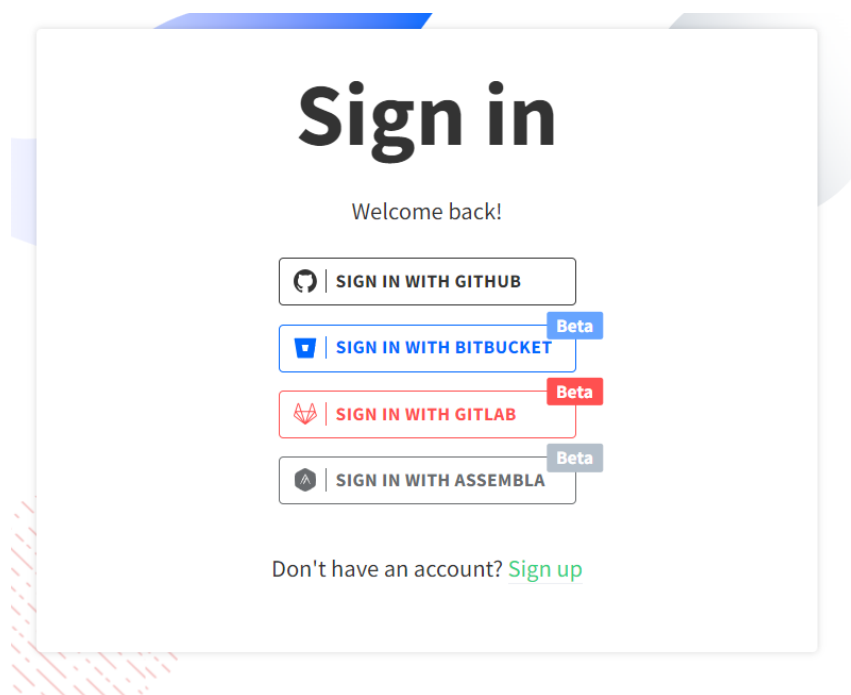
## Travis CI

Comenzamos integración con travis CI

El primer paso es ir a la página principal de travis CI

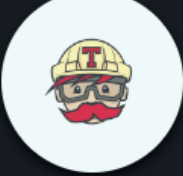


En la siguiente imagen podemos ver que travis nos pide conectar con algún repositorio, en nuestro caso github.






Travis nos pide permiso para conectar a nuestra cuenta de github y seleccionar un repositorio.



## Approve, Install, & Authorize Travis CI


Approve, Install, & Authorize on your personal account  
lisalvarez18 

suggested installation of this GitHub App now


for these repositories:

☐ **All repositories**  
This applies to all current *and* future repositories owned by the resource owner.  
Also includes public repositories (read-only).

☒ **Only select repositories**  
Select at least one repository.  
Also includes public repositories (read-only).

 **Select repositories** ▼

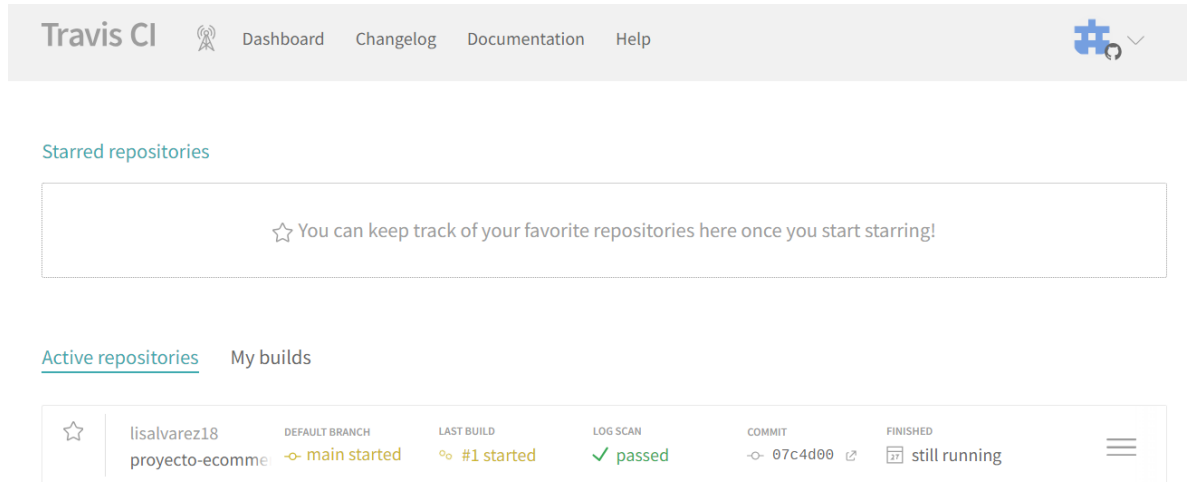
Search for a repository

 lisalvarez18/proyecto-e-commerce  
no description

with

✔

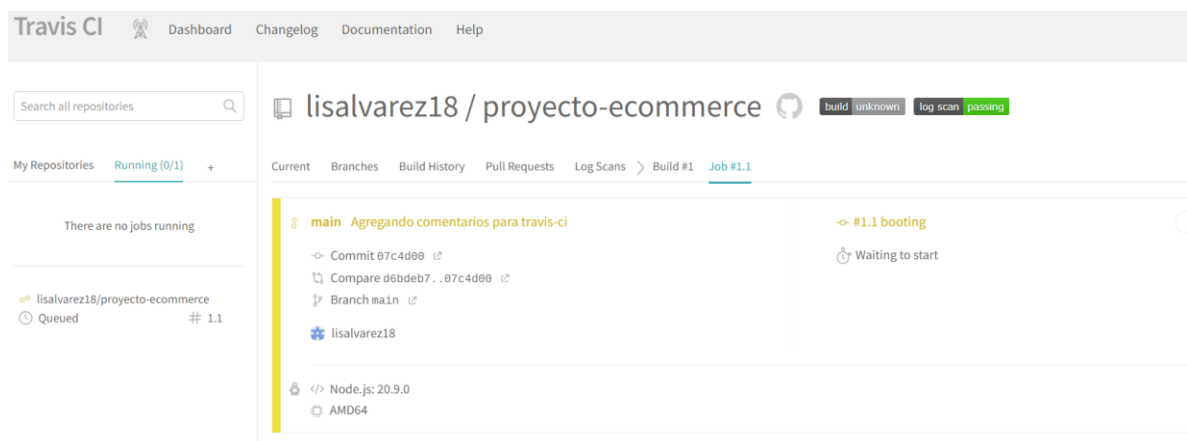
En el home de travis CI podemos ver que ya nuestro repositorio está plenamente integrado; aquí podemos ver información del último commit de nuestro Branch por defecto y del último scan.2.



The screenshot shows the Travis CI dashboard. At the top is a navigation bar with the Travis CI logo, a signal icon, and links for Dashboard, Changelog, Documentation, and Help. On the right is a user profile icon. Below the navigation bar, there's a section for 'Starred repositories' with a message: 'You can keep track of your favorite repositories here once you start starring!'. Below that is a section for 'Active repositories' and 'My builds'. The 'Active repositories' section shows a table with one repository: 'lisalvarez18 / proyecto-ecommerce'. The table has columns for repository name, default branch, last build, log scan, commit, and finished status. The 'main' branch is listed with a 'started' status, and the log scan is 'passed'. The commit is '07c4d00' and the build is 'still running'.

Repository	Default Branch	Last Build	Log Scan	Commit	Finished
lisalvarez18 / proyecto-ecommerce	main	#1 started	passed	07c4d00	still running

Aquí Podemos ver un scan en tiempo de ejecución el cual fue disparado por un commit, incluso podemos ver el mensaje del commit “Agregando comentarios para travis CI”.



The screenshot shows the Travis CI build page for the repository 'lisalvarez18 / proyecto-ecommerce'. The page has a navigation bar with the Travis CI logo, a signal icon, and links for Dashboard, Changelog, Documentation, and Help. Below the navigation bar, there's a search bar for repositories. The main content area shows the build details for the 'main' branch. The build is titled 'Agregando comentarios para travis-ci' and is in the 'booting' state. The commit is '07c4d00'. The build is triggered by a commit to the 'main' branch. The build is running on a Node.js 20.9.0 environment on an AMD64 architecture. The build is currently 'Waiting to start'.

Repository	Default Branch	Last Build	Log Scan	Commit	Finished
lisalvarez18 / proyecto-ecommerce	main	#1 started	passed	07c4d00	still running

Build #1.1

main Agregando comentarios para travis-ci

Commit 07c4d00

Compare d6bdeb7...07c4d00

Branch main

lisalvarez18

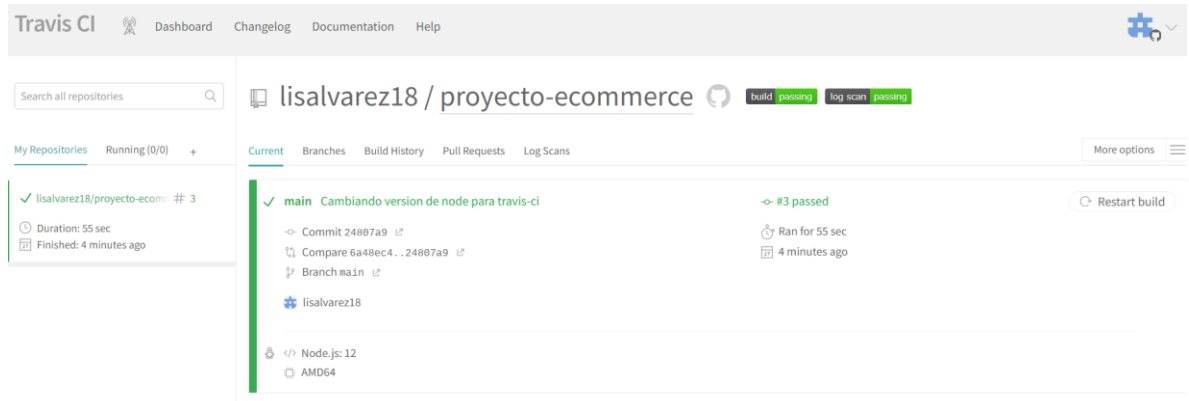
Node.js: 20.9.0

AMD64

#1.1 booting

Waiting to start

En este caso vemos otro scan que ha terminado de manera satisfactoria, esta vez este fue desencadenado por un commit en el cual se cambia la versión de node js para que el pipeline se ejecute correctamente.



Cuando seleccionamos el scan podemos observar el detalle de los logs arrojados durante la ejecución.

- En las líneas 1 y 7 podemos ver como travis prepara el sistema.
- En la línea 174 clona nuestro repositorio.
- En la línea 186 se instala la versión de node js que configuramos previamente, en este caso v12.
- En la línea 212 ejecuta el primer paso que definimos que es la instalación de dependencias.
- En la línea 331 ejecuta el segundo paso que definimos que es la ejecución de tests.

Remove logRaw log

1 Worker information0.07s

6

7 Build system information0.01s

171

172 OK0.32s

1732.35s

174\$ git clone --depth=50 --branch=main https://github.com/lisalvarez18/proyecto-ecommerce.git lisalvarez18/proyecto-ecommerce0.65s

184

185

186\$ npm install120.01s

1923.02s

193 Setting up build cache

201

202

205\$ node --version

206v12.22.12

207\$ npm --version

2086.14.16

209\$ npm --version

2100.39.5

211

212\$ npm installinstall.npm10.35s

316

317\$ echo 'Installing dependencies'

319\$ npm installbefore\_script.10.09s

320\$ echo 'Running tests' - npm testbefore\_script.22.79s

321Running tests - npm test0.09s

322The command "echo 'Running tests' - npm test" exited with 0.

323store build cachecache.2

327

328

329Done. Your build exited with 0.

Top

Aquí Podemos observar el historial de ejecuciones en travis, donde las 2 primeras fueron fallidas y la más reciente se ejecutó correctamente.

lisalvarez18 / proyecto-ecommerce

build passinglog scan passing

Current

Branches

Build History

Pull Requests

Log Scans

More options

Default Branch

✓ main

# 3 passed

24807a9

✓

!

!

3 builds

18 minutes ago

lisalvarez18

lisalvarez18 / proyecto-ecommerce

build passinglog scan passing

Current

Branches

Build History

Pull Requests

Log Scans

More options

✓ main

lisalvarez18

Cambiando version de node para travis-ci

#3 passed

55 sec

19 minutes ago

24807a9

! main

lisalvarez18

Cambiando version de node para travis-ci

#2 errored

37 sec

22 minutes ago

6a48ec4

! main

lisalvarez18

Agregando comentarios para travis-ci

#1 errored

34 sec

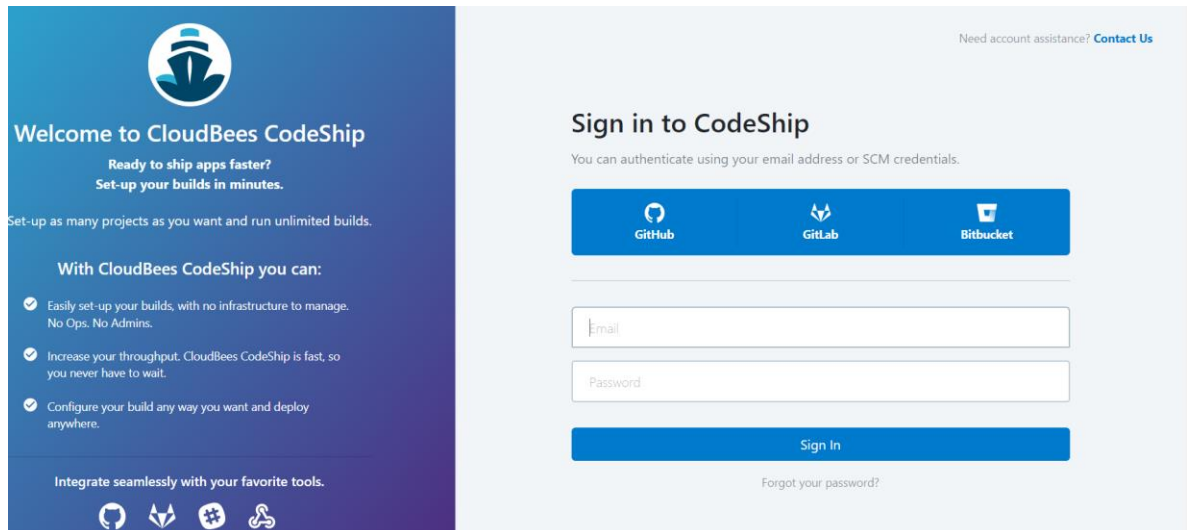
28 minutes ago

07c4d00

Show more

## CodeShip




Para empezar con la configuración vamos a la página principal de codeship el cual nos invita a ingresar usando un sistema de repositorio.




The image shows the CloudBees CodeShip landing page and a sign-in form. The landing page on the left has a blue gradient background with the CodeShip logo at the top. It includes the text "Welcome to CloudBees CodeShip", "Ready to ship apps faster? Set-up your builds in minutes.", and "Set-up as many projects as you want and run unlimited builds." Below this, it lists benefits: "With CloudBees CodeShip you can:" followed by three bullet points: "Easily set-up your builds, with no infrastructure to manage. No Ops. No Admins.", "Increase your throughput. CloudBees CodeShip is fast, so you never have to wait.", and "Configure your build any way you want and deploy anywhere." At the bottom, it says "Integrate seamlessly with your favorite tools." and shows icons for GitHub, GitLab, Jenkins, and Bitbucket.

The sign-in form on the right has a light blue background. It includes a link "Need account assistance? [Contact Us](#)" at the top right. The main heading is "Sign in to CodeShip" with the subtext "You can authenticate using your email address or SCM credentials." Below this are three buttons for authentication: "GitHub", "GitLab", and "Bitbucket". There are two input fields: "Email" and "Password". A "Sign in" button is at the bottom. A link "Forgot your password?" is at the bottom right.


Codeship nos pide permisos para conectarse con nuestro repositorio.




CloudBees by **CloudBees** would like permission to:




Verify your GitHub identity (lisalvarez18)



Know which resources you can access




Act on your behalf



[Learn more](#)

**Resources on your account**

---



**Email addresses** (read)  
View your email addresses

---

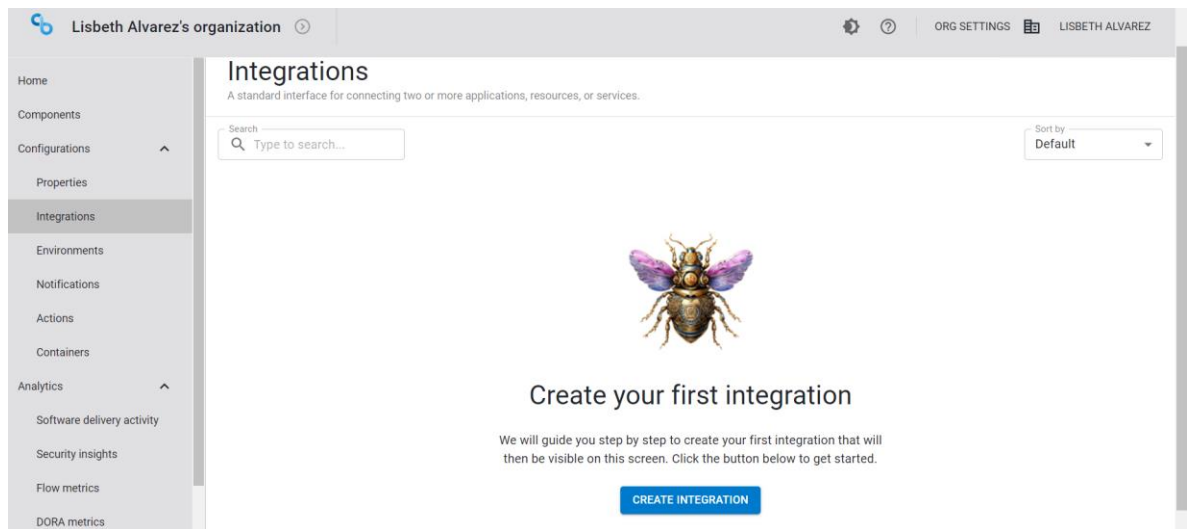
[Learn more about CloudBees](#)

Cancel


Authorize CloudBees

Authorizing will redirect to  
<https://app.codeship.com>


En el panel Izquierdo vamos a la sección de integraciones y seleccionamos “Crear integración”.



A continuación cloudbees nos pide permisos para ingresar al repositorio de nuestro proyecto de ecommerce.




## Install & Authorize CloudBees Platform

Install & Authorize on your personal account lisalvarez18 


for these repositories:

☐ **All repositories**  
This applies to all current *and* future repositories owned by the resource owner.  
Also includes public repositories (read-only).

☒ **Only select repositories**  
Select at least one repository.  
Also includes public repositories (read-only).

 **Select repositories** ▾

Search for a repository

 lisalvarez18/proyecto-ecommerce  
no description

with

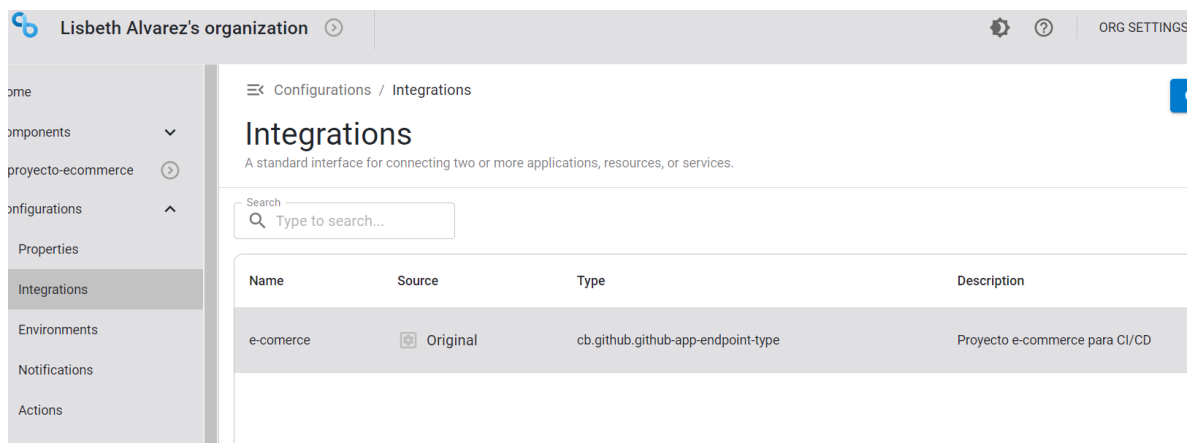
✓ Read and write access to metadata and repository contents

✓ **Read and write** access to code, commit statuses, and pull requests

**Install & Authorize** [Cancel](#)



Aquí Podemos ver la integración que acabamos de crear para nuestro repositorio.



The screenshot shows the GitHub 'Integrations' page for the organization 'Lisbeth Alvarez's organization'. The left sidebar contains a navigation menu with items: 'ome', 'omponents', 'proyecto-e-commerce', 'onfigurations', 'Properties', 'Integrations' (highlighted), 'Environments', 'Notifications', and 'Actions'. The main content area is titled 'Integrations' and includes a search bar with the placeholder 'Type to search...'. Below the search bar is a table with the following data:

Name	Source	Type	Description
e-commerce	Original	cb.github.github-app-endpoint-type	Proyecto e-commerce para CI/CD

A continuación, procedemos a crear un componente usando la integración creada en el paso anterior.

code.

Trigger summary

Create component

×

Search

🔍 Type to search...

☐ Hide connected repositories

🔗

lisalvarez18/proyecto-ecommerce

🐙

ⓘ

Repository not listed? Contact an administrator to connect your repository.

CONNECT YOUR REPOSITORY

Configuration branch

Luego Podemos apreciar que el proyecto es creado de manera satisfactoria.

☰ Components

CREATE COMPONENT

Components

A component is a piece of software, built and, if desired, released and deployed from source code.

FILTER

Search

🔍 Type to search...

Sort by

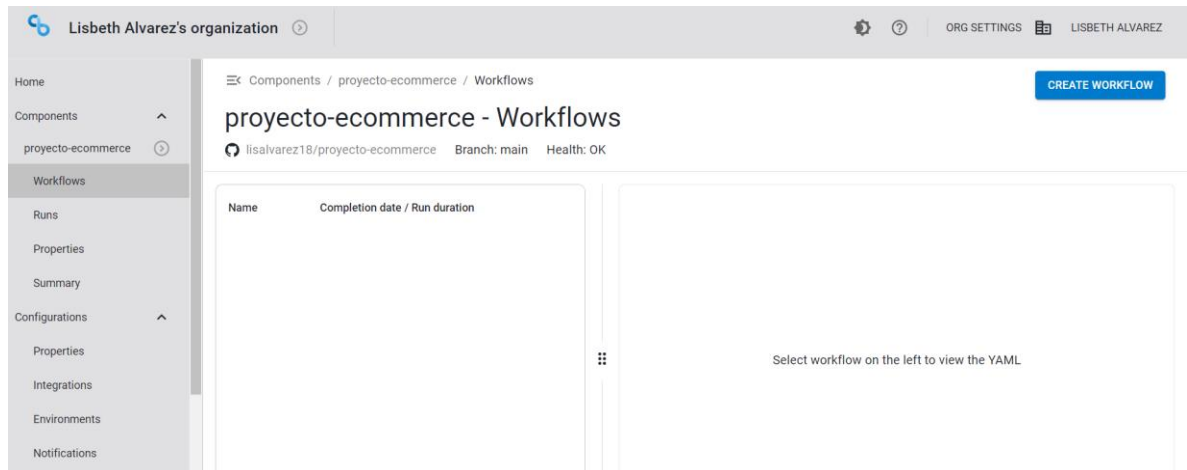
Default

⌵

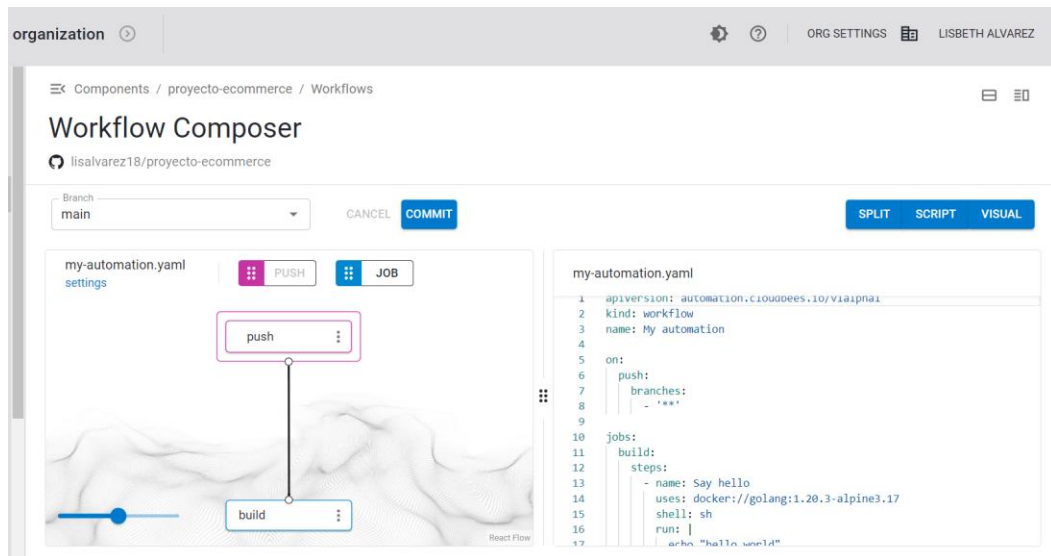
⌵

Component	Trigger summary	Last run
<div>🔍 proyecto-ecommerce</div> <div>main</div>	-	-

Seleccionamos nuestro componente y allí seleccionamos el botón “Crear flujo de trabajo”.



En esta Ventana tenemos dos herramientas para crear y editar nuestros flujos de trabajo, en el de la izquierda podemos usar componentes y arrastrarlos y acomodarlos según nuestra necesidad, mientras que a la derecha tenemos un editor para escribir nuestras configuraciones en formato yaml. Por defecto nos carga un flujo de trabajo que imprime un mensaje “Hola mundo”.



Esta es nuestra configuración para el flujo de trabajo de nuestra aplicación.

- En la línea 11 definimos el job.
- Definimos 3 pasos, el primero es check out en la línea 16, el cual descarga nuestro código de nuestro repositorio github.

El segundo paso es la instalación de dependencias en la línea 18.

El tercer paso es la ejecución de pruebas en la línea 22.

- En las líneas 19 y 23 definimos la versión de node js que usaremos para la instalación de dependencias y las pruebas.

```
my-automation.yaml
10 jobs:
11   ci-job:
12     steps:
13       - uses: docker://alpine/git:latest
14         run: |
15           git config --global --add safe.directory /cloudbees/workspac
16       - name: checkout
17         uses: cloudbees-io/checkout@v1
18       - name: Build node app
19         uses: docker://node:20.9.0-alpine3.18
20         run: |
21           npm install
22       - name: Run tests
23         uses: docker://node:20.9.0-alpine3.18
24         run: |
25           npm test
```

Luego de definir nuestra configuración, seleccionamos el botón commit y agregamos un nombre al archivo que contendrá la configuración y un comentario para el commit; finalmente seleccionamos finish.

**Commit workflow** ✕

Commit information

File name \*  
my-automation.yaml

Commit message \*  
Agregando configuracion para cloudbees-codeship

Select a branch to commit

☒ Commit to a current branch: main

CANCEL FINISH

Background content:

ows

CANCEL COMMIT

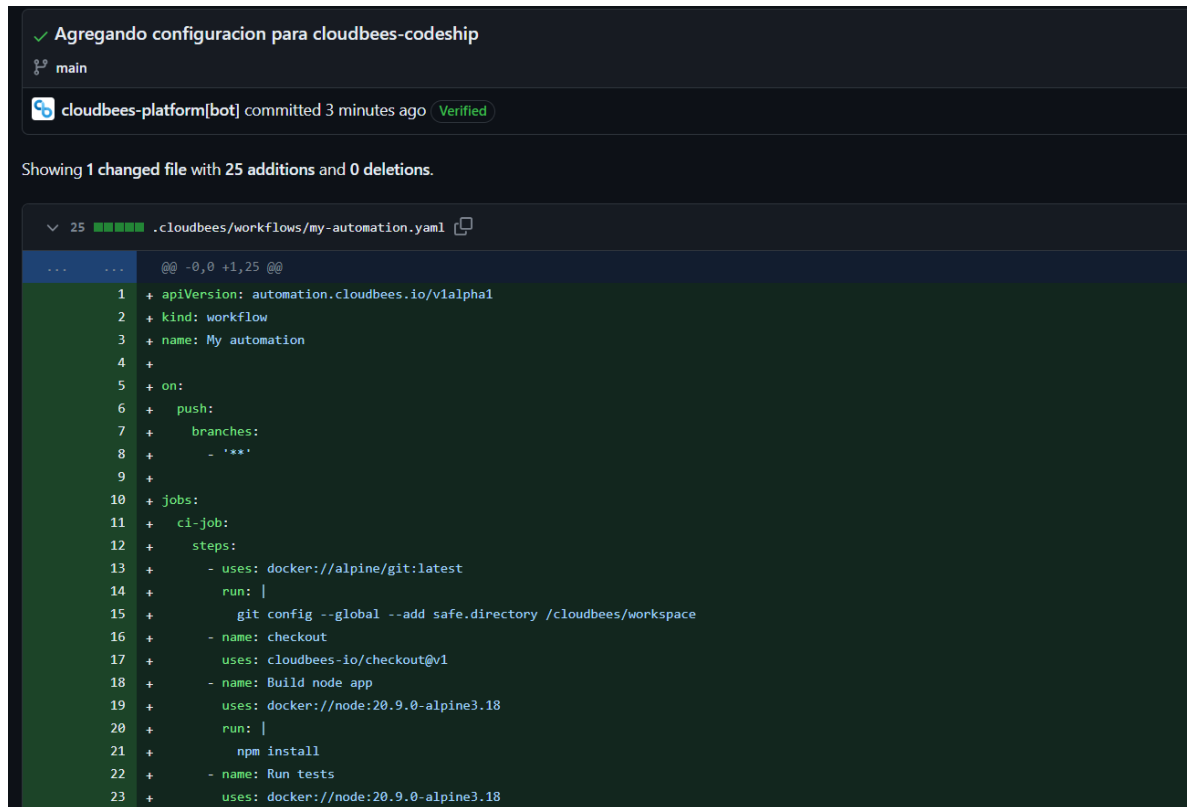
JOB

my-automation.yaml

```
10 jobs:
11   ci-job:
12     steps:
13       - uses: dock
14         run: |
15           git conf
16       - name: chec
17         uses: clou
18       - name: Buil
19         uses: dock
20         run: |
21           npm inst
22       - name: Run
23         uses: dock
24         run: |
25           npm test
```

React Flow

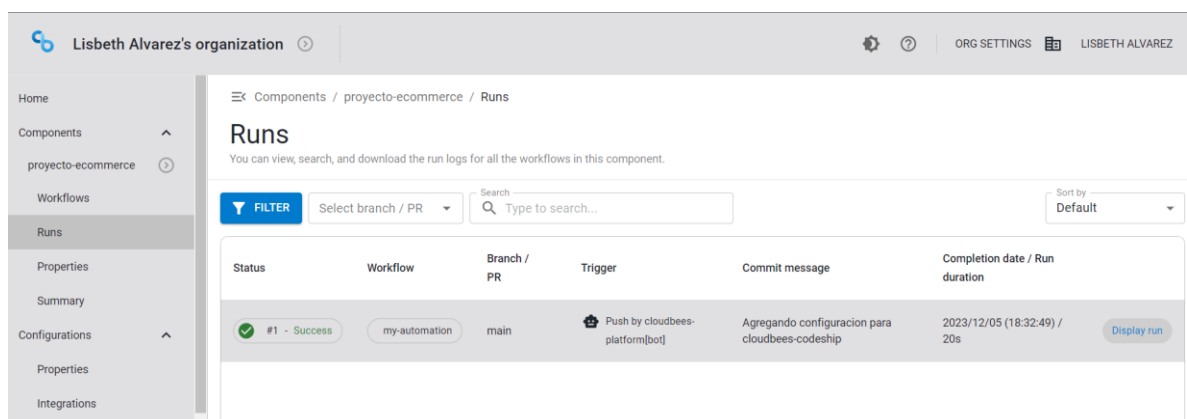
Podemos observar que esta configuración se sube a nuestro repositorio de git hub en la carpeta .cloudbees/workflows/my-automation.yaml



The screenshot shows a GitHub commit interface. At the top, it says "✓ Agregando configuracion para cloudbees-codeship" and "main". Below that, it says "cloudbees-platform[bot] committed 3 minutes ago" with a "Verified" badge. The commit message is "Showing 1 changed file with 25 additions and 0 deletions." The file is ".cloudbees/workflows/my-automation.yaml". The code is as follows:

```
@@ -0,0 +1,25 @@
1 + apiVersion: automation.cloudbees.io/v1alpha1
2 + kind: workflow
3 + name: My automation
4 +
5 + on:
6 +   push:
7 +     branches:
8 +       - '*'
9 +
10 + jobs:
11 +   ci-job:
12 +     steps:
13 +       - uses: docker://alpine/git:latest
14 +         run: |
15 +           git config --global --add safe.directory /cloudbees/workspace
16 +       - name: checkout
17 +         uses: cloudbees-io/checkout@v1
18 +       - name: Build node app
19 +         uses: docker://node:20.9.0-alpine3.18
20 +         run: |
21 +           npm install
22 +       - name: Run tests
23 +         uses: docker://node:20.9.0-alpine3.18
```

Este commit dispara un scan con el flujo de trabajo que definimos anteriormente. Esto lo podemos observar en la sección runs.



The screenshot shows the CloudBees UI for "Lisbeth Alvarez's organization". The left sidebar has a menu with "Home", "Components", "projecto-ecommerce", "Workflows", "Runs", "Properties", "Summary", "Configurations", "Properties", and "Integrations". The main content area is titled "Components / proyecto-ecommerce / Runs". Below the title, it says "You can view, search, and download the run logs for all the workflows in this component." There is a "FILTER" button, a "Select branch / PR" dropdown, a search bar, and a "Sort by" dropdown set to "Default". The table below shows the runs:

Status	Workflow	Branch / PR	Trigger	Commit message	Completion date / Run duration
✓ #1 - Success	my-automation	main	Push by cloudbees-platform[bot]	Agregando configuracion para cloudbees-codeship	2023/12/05 (18:32:49) / 20s <a href="#">Display run</a>

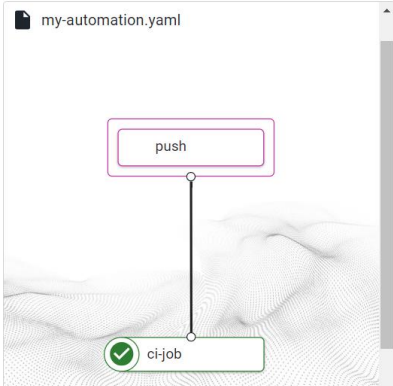
Cuando seleccionamos la ejecución podemos ver el detalle de cada paso durante la ejecución.

Components / proyecto-ecommerce / Runs / main / 97e71aa0

RERUN WORKFLOW STOP RUN

### Run detail

Status: **SUCCESS** Start date: 2023/12/05 (18:32:29) End date: 2023/12/05 (18:32:49) Duration: 20s



my-automation.yaml

```
graph TD; push[push] --> ci-job[ci-job];
```

Step	Start	Duration
✓ ci-job	Start: 18:32:29	Duration: 20s
✓ Set up job		6s
✓ Run script		
✓ Checkout		
✓ Build node app		10s
✓ Run tests		2s

Cuando seleccionamos cada paso podemos ver los logs arrojados por cada paso. En este caso podemos ver el paso para instalar las dependencias.

Search Type to search EXPAND ALL

^ ✓ Build node app GO TO END 10s

```
1
2 added 373 packages, and audited 374 packages in 8s
3
4 44 packages are looking for funding
5   run `npm fund` for details
6
7 found 0 vulnerabilities
8 npm notice
9 npm notice New minor version of npm available! 10.1.0 -> 10.2.4
10 npm notice Changelog: <https://github.com/npm/cli/releases/tag/v10.2.4>
11 npm notice Run `npm install -g npm@10.2.4` to update!
12 npm notice
```

En este caso Podemos ver la ejecución de las pruebas.

Search

🔍 Type to search

EXPAND ALL

⋮

^

✓ Run tests

GO TO END 2s

```
detectOpenHandles --forceExit
4
5 PASS ./test.js
6   Homepage
7     ✓ GET / return Hola mundo (131 ms)
8     ✓ GET /index.html return Hola mundo (10 ms)
9
10 Test Suites: 1 passed, 1 total
11 Tests:      2 passed, 2 total
12 Snapshots:  0 total
13 Time:       0.803 s
14 Ran all test suites.
```

## Historial de cambios

### first commit

 [lsalvarez18](#) created this branch • 55907e1 •


28 days ago

### Agregando configuracion de docker y hola mundo

 [lsalvarez18](#) pushed 1 commit • 55907e1...e13250e •

27 days ago


### Agregando documentacion entrega 1

 [lsalvarez18](#) pushed 1 commit • e13250e...fa26caa •

27 days ago




## Adding Jenkinsfile

 [lisalvarez18](#) pushed 1 commit • fa26caa...3def468 •

19 days ago

## Adding test stage to jenkinsfile, adding gitignore, adding customized...

 [lisalvarez18](#) pushed 1 commit • 3def468...e8dc019 •

15 days ago

## Moviendo jenkins file, modificando script para test

 [lisalvarez18](#) pushed 1 commit • e8dc019...da6a399 •


15 days ago

## Modificando jenkinsfile

 [lisalvarez18](#) pushed 1 commit • da6a399...a9ffaf6 •

14 days ago

## Cambiando la version de node

 [lisalvarez18](#) pushed 1 commit • a9ffaf6...7966ab0 •


14 days ago

## Borrando comando npm cache clear

 [lisalvarez18](#) pushed 1 commit • 7966ab0...2282409 •


14 days ago

## Agregando Deliver stage

 [lisalvarez18](#) pushed 1 commit • 2282409...e466a87 •

14 days ago

## Borrando npm build

 [lisalvarez18](#) pushed 1 commit • e466a87...28475dc •

14 days ago

## agregando entrega 2



[lisalvarez18](#) pushed 1 commit • 28475dc...ef98cba •

14 days ago

## Agregando configuracion para travis-ci



[lisalvarez18](#) pushed 1 commit • ef98cba...d6bdeb7 •

6 days ago

## Agregando comentarios para travis-ci



[lisalvarez18](#) pushed 1 commit • d6bdeb7...07c4d00 •

6 days ago

## Cambiando version de node para travis-ci



[lisalvarez18](#) pushed 1 commit • 07c4d00...6a48ec4 •

6 days ago

## Cambiando version de node para travis-ci



[lisalvarez18](#) pushed 1 commit • 6a48ec4...24807a9 •

6 days ago

## Agregando configuracion para cloudbees-codeship



[cloudbees-platform\[bot\]](#) pushed 1 commit • 24807a9...37762dc •

2 hours ago