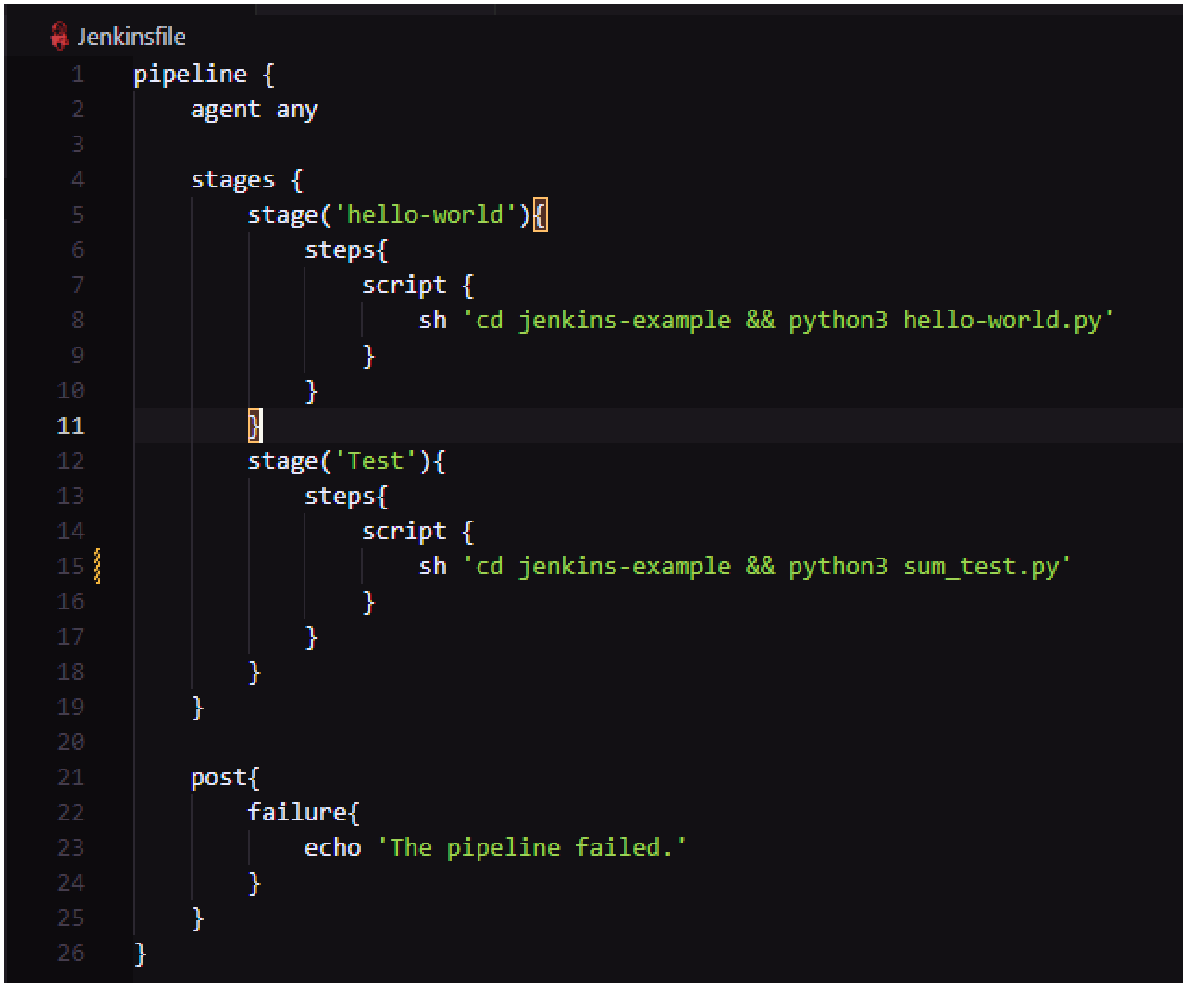
Creating a simple CI with Jenkins

Create a new Jenkinsfile in the root folder and create a pipeline:



Jenkinsfile

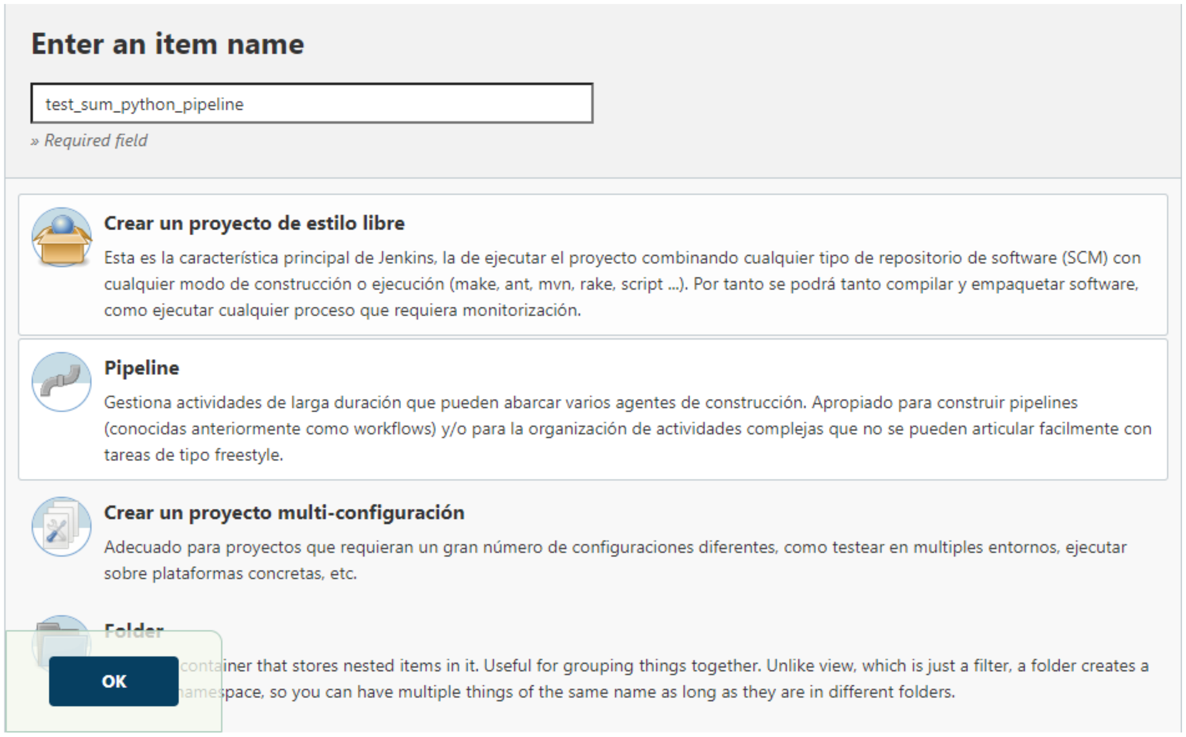
```
pipeline {
    agent any

    stages {
        stage('hello-world'){
            steps{
                script {
                    sh 'cd jenkins-example && python3 hello-world.py'
                }
            }
        }
        stage('Test'){
            steps{
                script {
                    sh 'cd jenkins-example && python3 sum_test.py'
                }
            }
        }
    }

    post{
        failure{
            echo 'The pipeline failed.'
        }
    }
}
```

Configuring Jenkins Job to connect with GitHub

Create a new Job and select Pipeline. Add a descriptive name.



Enter an item name

test_sum_python_pipeline

» *Required field*

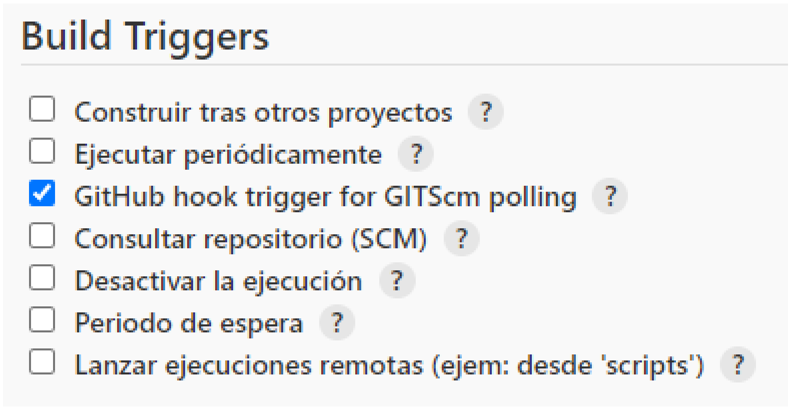
Crear un proyecto de estilo libre
Esta es la característica principal de Jenkins, la de ejecutar el proyecto combinando cualquier tipo de repositorio de software (SCM) con cualquier modo de construcción o ejecución (make, ant, mvn, rake, script ...). Por tanto se podrá tanto compilar y empaquetar software, como ejecutar cualquier proceso que requiera monitorización.

Pipeline
Gestiona actividades de larga duración que pueden abarcar varios agentes de construcción. Apropiado para construir pipelines (conocidas anteriormente como workflows) y/o para la organización de actividades complejas que no se pueden articular facilmente con tareas de tipo freestyle.

Crear un proyecto multi-configuración
Adecuado para proyectos que requieran un gran número de configuraciones diferentes, como testear en multiples entornos, ejecutar sobre plataformas concretas, etc.

Folder
container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a namespace, so you can have multiple things of the same name as long as they are in different folders.

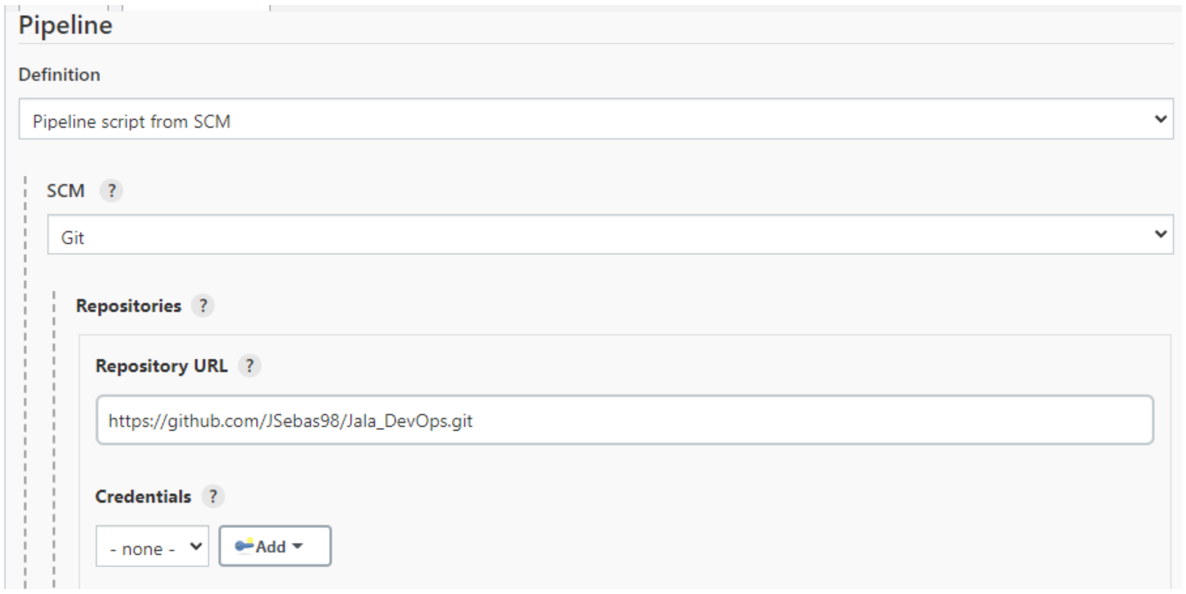
Choose “Github hook trigger for GITScm polling” option.



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') ?

In the Pipeline block choose “Pipeline script from SCM,” then choose “Git” in SCM and add the link to the GitHub repository in Repository URL.



Pipeline

Definition: Pipeline script from SCM

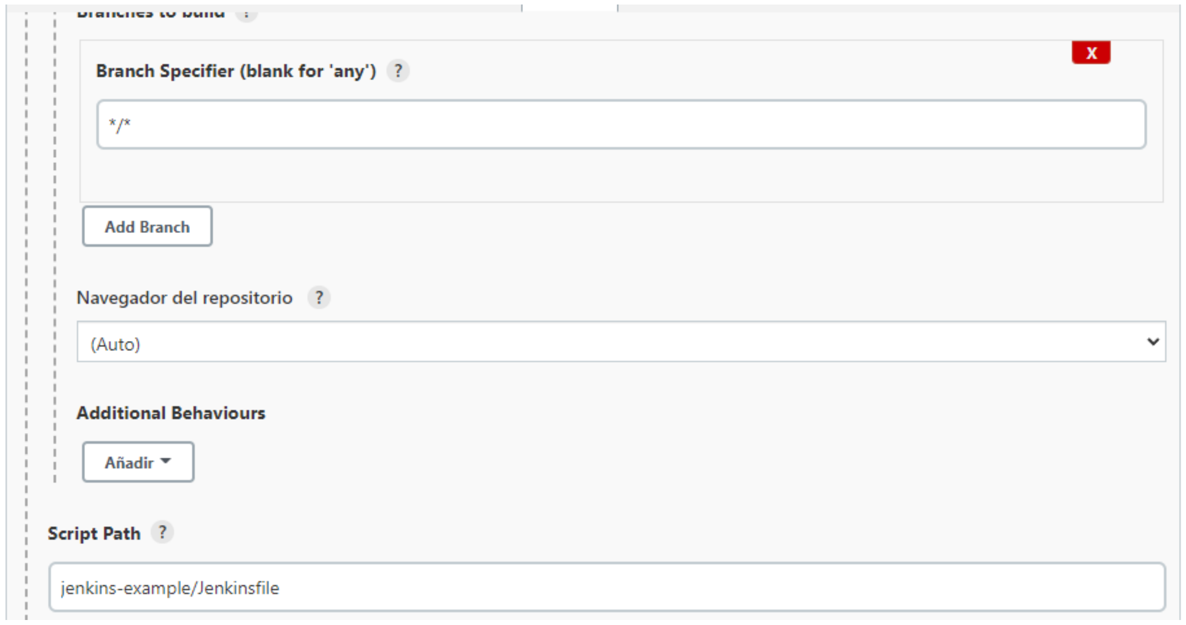
SCM: Git

Repositories

Repository URL: https://github.com/JSebas98/Jala_DevOps.git

Credentials: - none -

Write "*/*" in the Branch Specifier blank so that Jenkins will check all branches.



Branch Specifier (blank for 'any'): */*

Navegador del repositorio: (Auto)

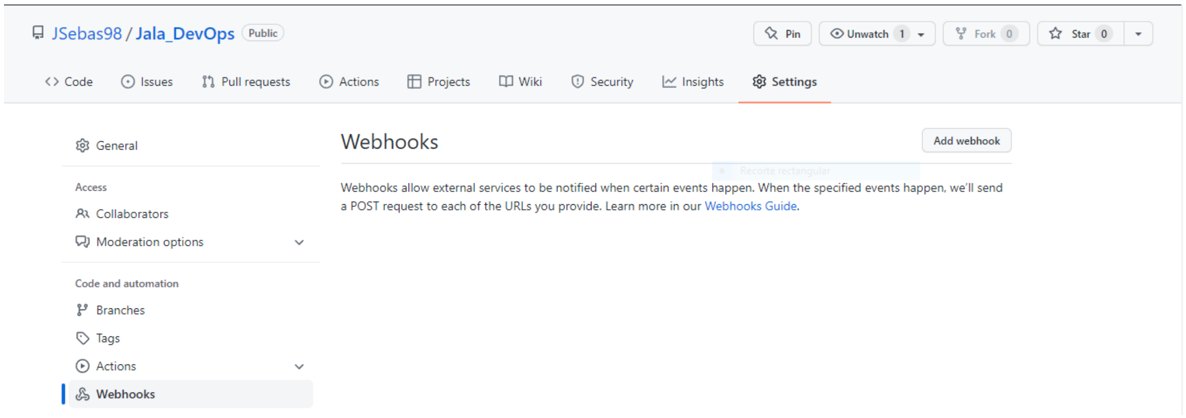
Additional Behaviours

Script Path: jenkins-example/Jenkinsfile

If the Jenkinsfile is not on the root directory in the repository, make sure to add the right path to it in Script Path.

Configuring GitHub to connect with Jenkins

Create a Webhook in your GitHub repository by going to Settings/Webhooks/Add webhook:

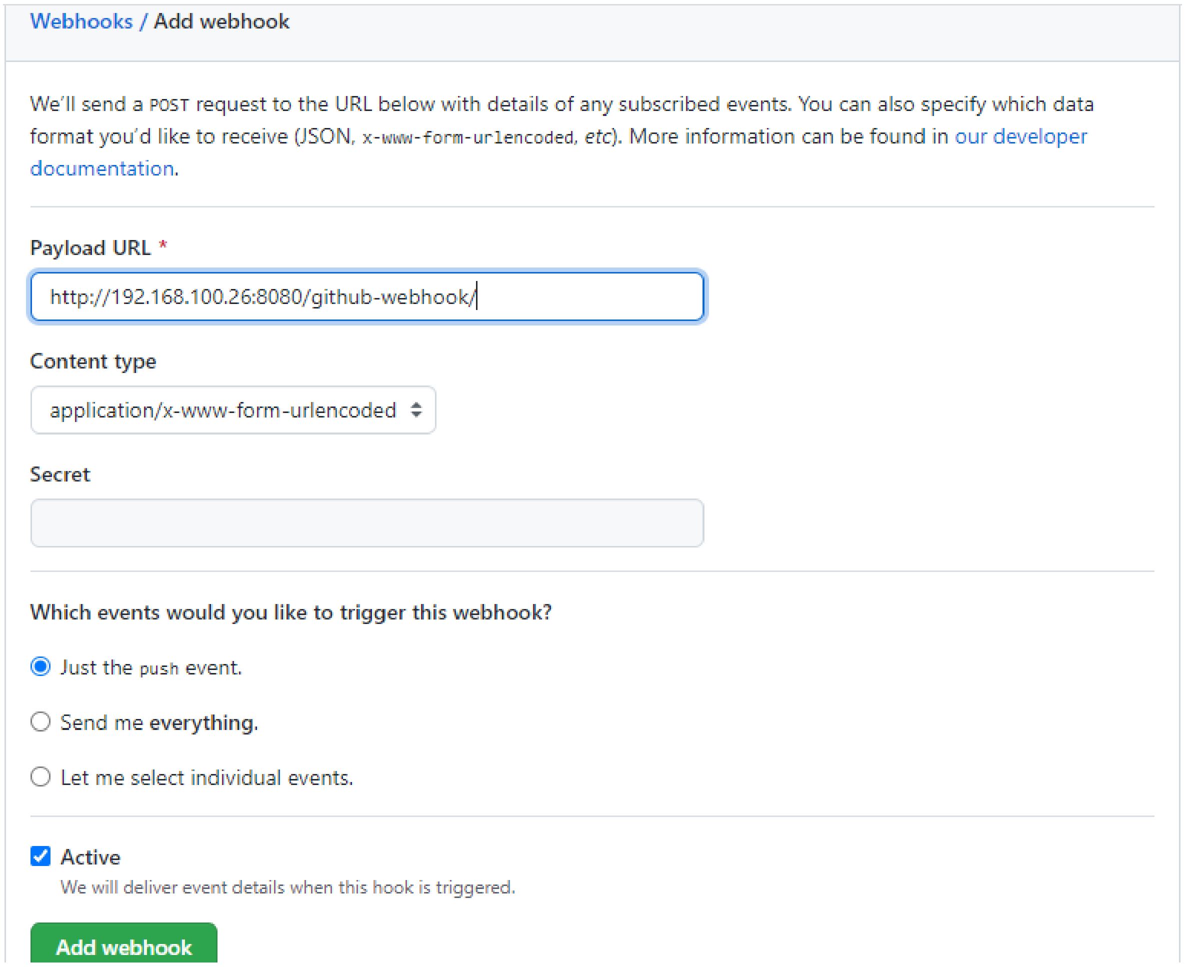


JSebas98 / Jala_DevOps Public

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

Add Jenkins URL (generally <your_machine_IP:<port>) followed by /github-webhook/ in Payload URL. Make sure to select the option “Just the push event” and mark the “Active” option. Then click on “Add webhook”.



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 *: http://192.168.100.26:8080/github-webhook/

Content type: application/x-www-form-urlencoded

Secret:

Which events would you like to trigger this webhook?

[✓ Just the `push` event.]

Send me **everything**.

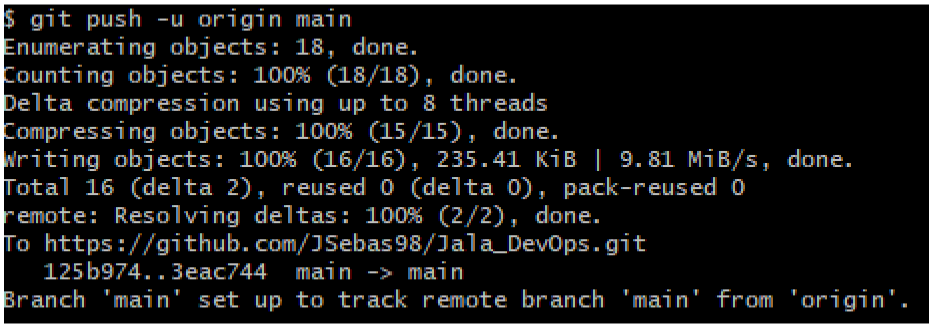
Let me select individual events.

[✓ Active]

We will deliver event details when this hook is triggered.

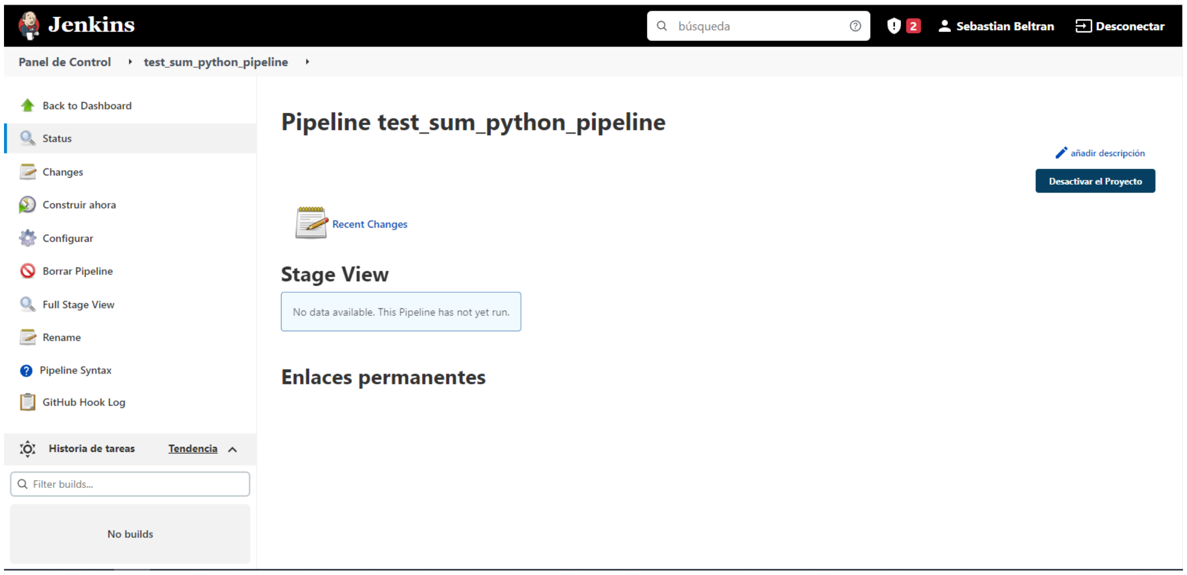
Testing the connection

Make a commit to the GitHub repository



```
$ git push -u origin main
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (16/16), 235.41 KiB | 9.81 MiB/s, done.
Total 16 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/JSebas98/Jala_DevOps.git
   125b974..3eac744  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Access the job dashboard in Jenkins



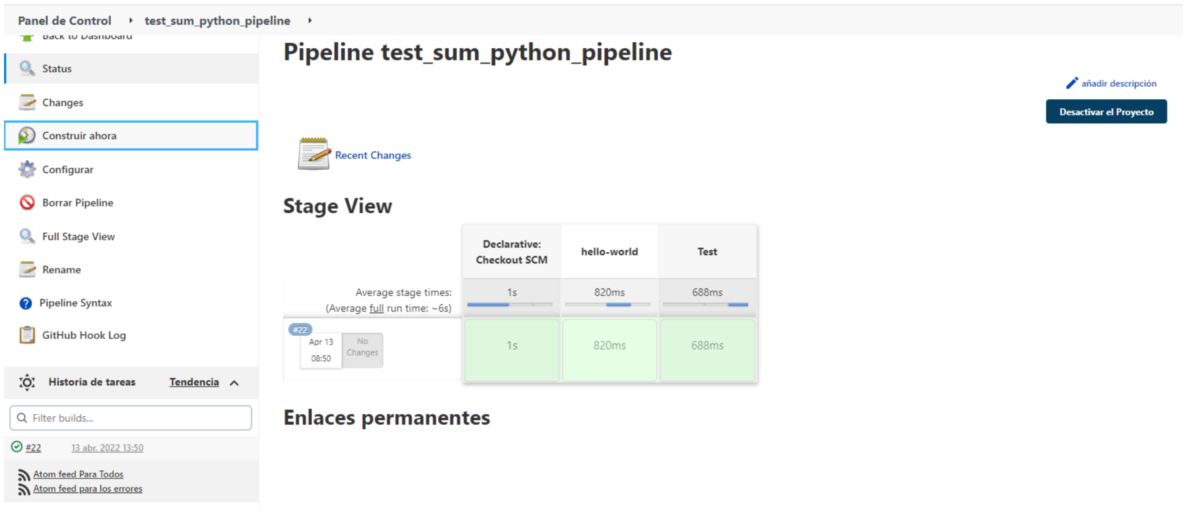
Pipeline test_sum_python_pipeline

Stage View

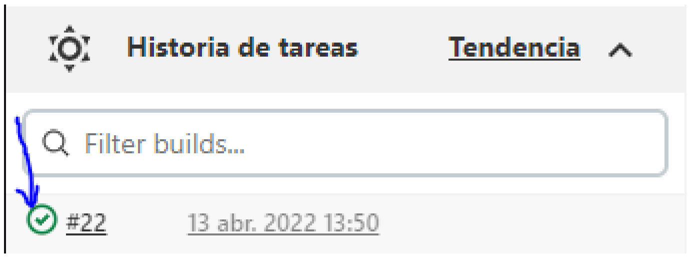
No data available. This Pipeline has not yet run.

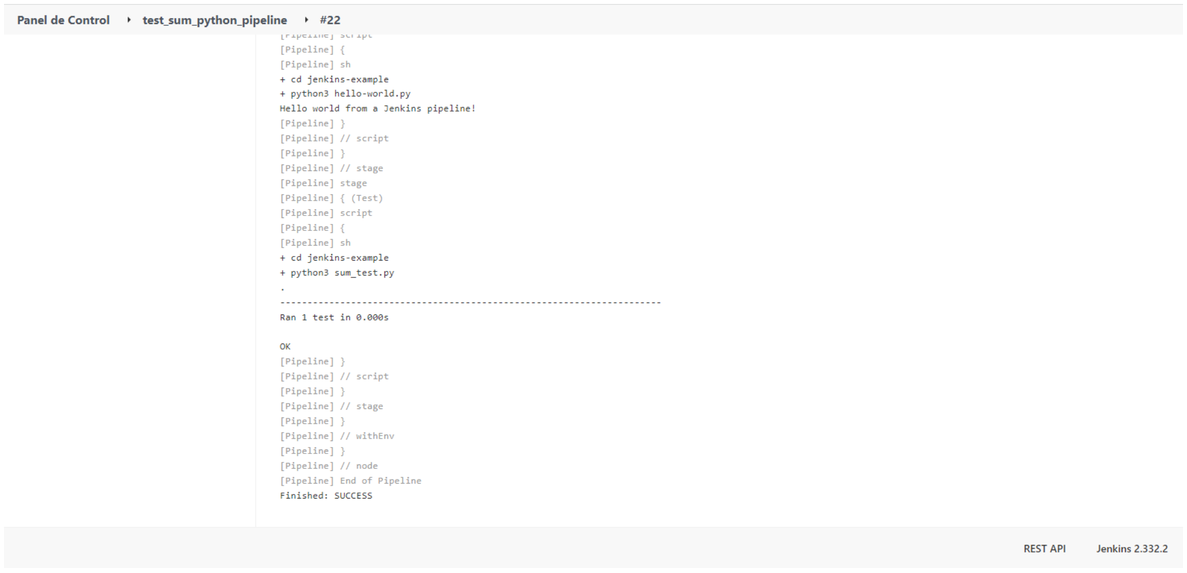
Enlaces permanentes

Build the job by clicking on "Build now". If everything is configured right, all stages should be green.



Inside the “Build History” there should be the record of the executed job. Click on the green tick (or the red cross if there is any error) to check the console output.





```
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ cd jenkins-example
+ python3 hello-world.py
Hello world from a Jenkins pipeline!
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ cd jenkins-example
+ python3 sum_test.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

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