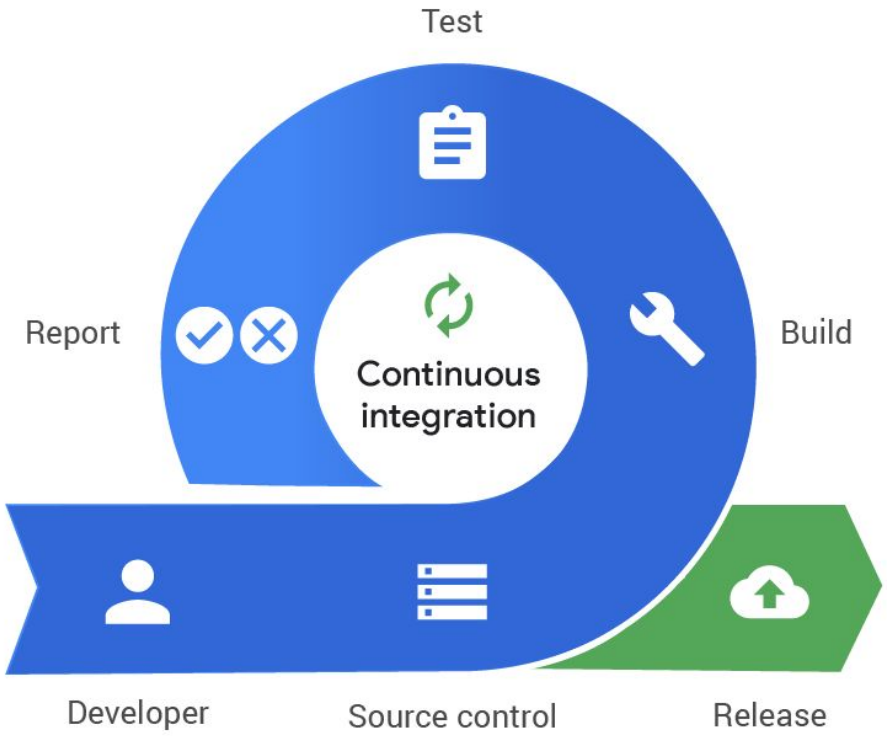
Continuous Integration (CI)

Alejandro García Perdomo (alu0101312101)
Jorge Cabrera Rodríguez (alu0101351773)

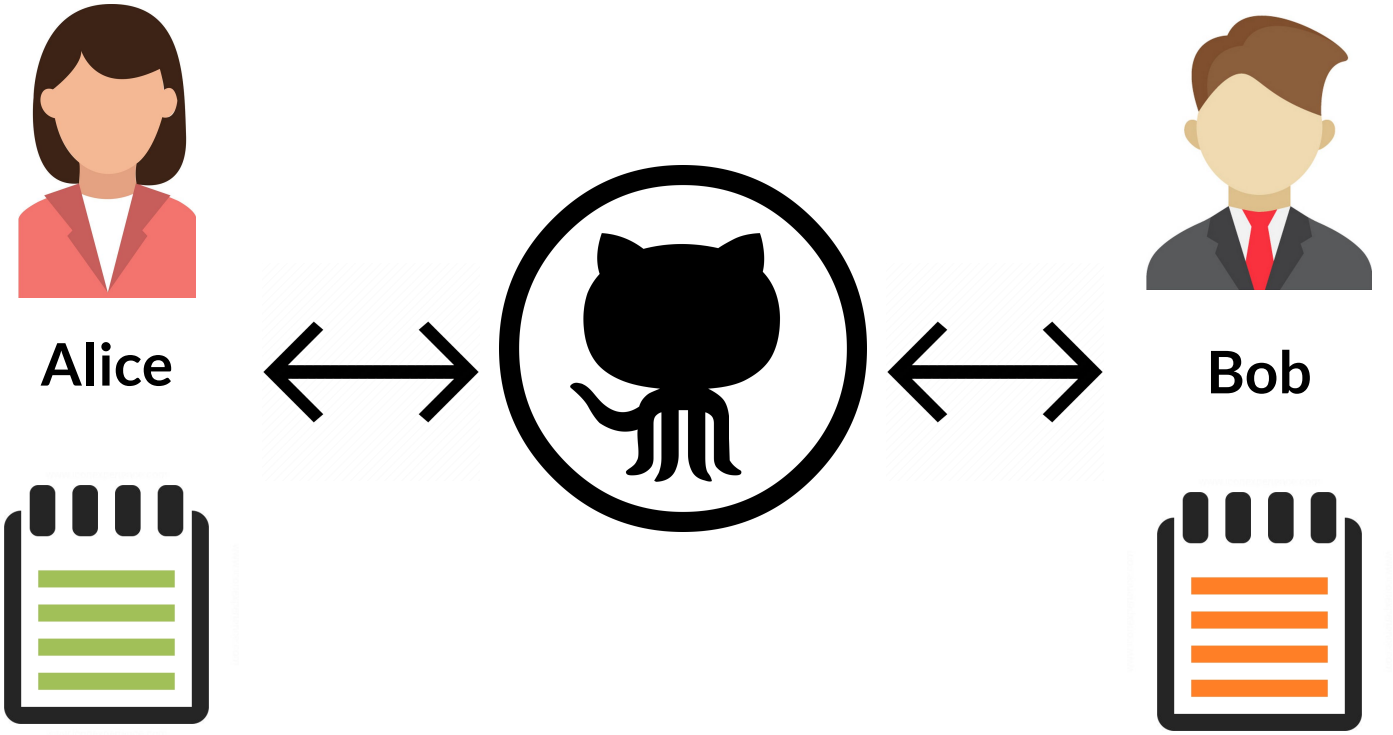
Tips to be covered

1. Introduction
2. What is CI
3. CI vs CD vs CDel
4. Phases of CI
5. CircleCI
    a. Docker
    b. Steps
    c. Jobs
    d. Workflow
    e. Relevant features

A brief introduction to CI



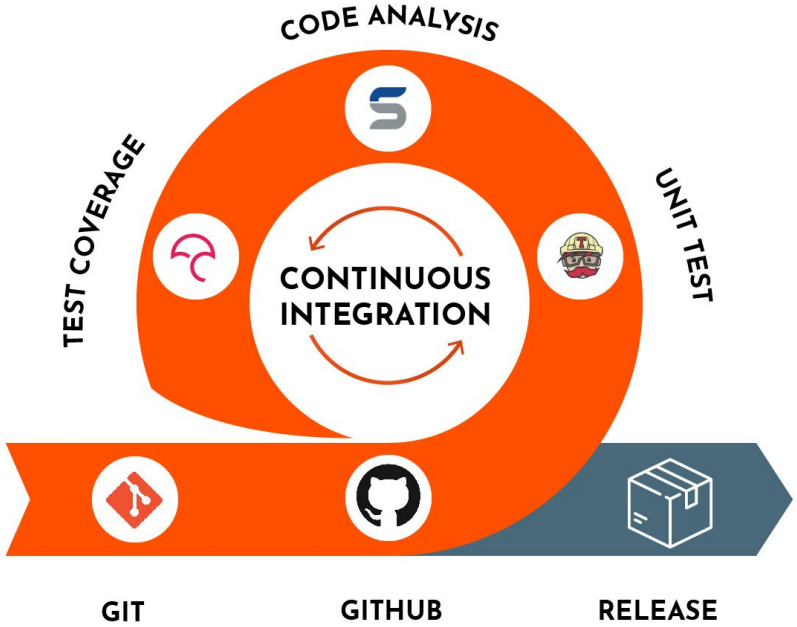
An introduction example



Push frequently, save lives

1. git commit
2. git push
3. leave building

Use CI!


CODE ANALYSIS
TEST COVERAGE
UNIT TEST
CONTINUOUS INTEGRATION
GIT
GITHUB
RELEASE

What is CI?

Development practice where developers **integrate code** into a shared repository **frequently.**

Each integration can then be verified by an **automated build** and **automated tests**.

Advantages and Disadvantages

Advantages

- Merging is much easier.
- There is always executable code.
- Bugs are easier to find.
- More communication between team members.

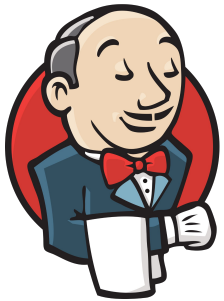
Disadvantages

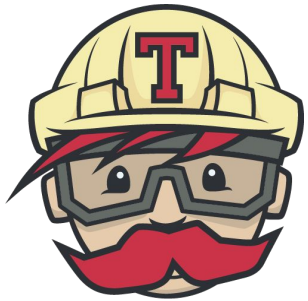
- May be slower.
- Needs a lot of push and pull actions.

“Continuous Integration doesn’t get rid of bugs, but it does make them dramatically easier to find and remove.”

-Martin Fowler, software developer-

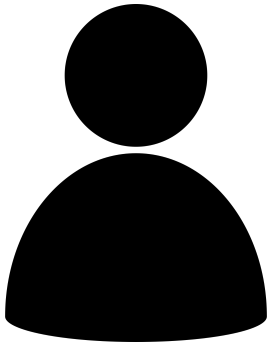
CI is independent of software





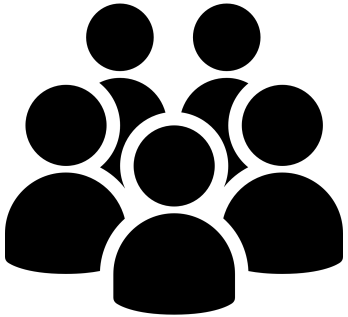
Practical application of CI

Small projects



- Might help with development
- Very situational
- Not required

Big projects

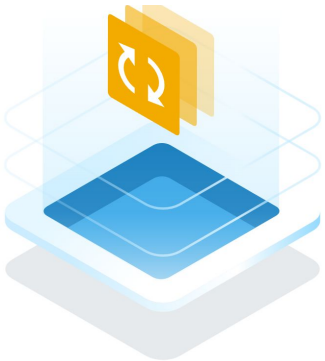


- Mandatory for teamwork
- Requires an administrator
- Make the projects success

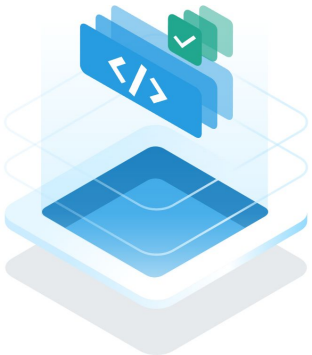
Continuous Integration
vs
Continuous Deployment
vs
Continuous Delivery

Differences between each model

Continuous Integration (CI)

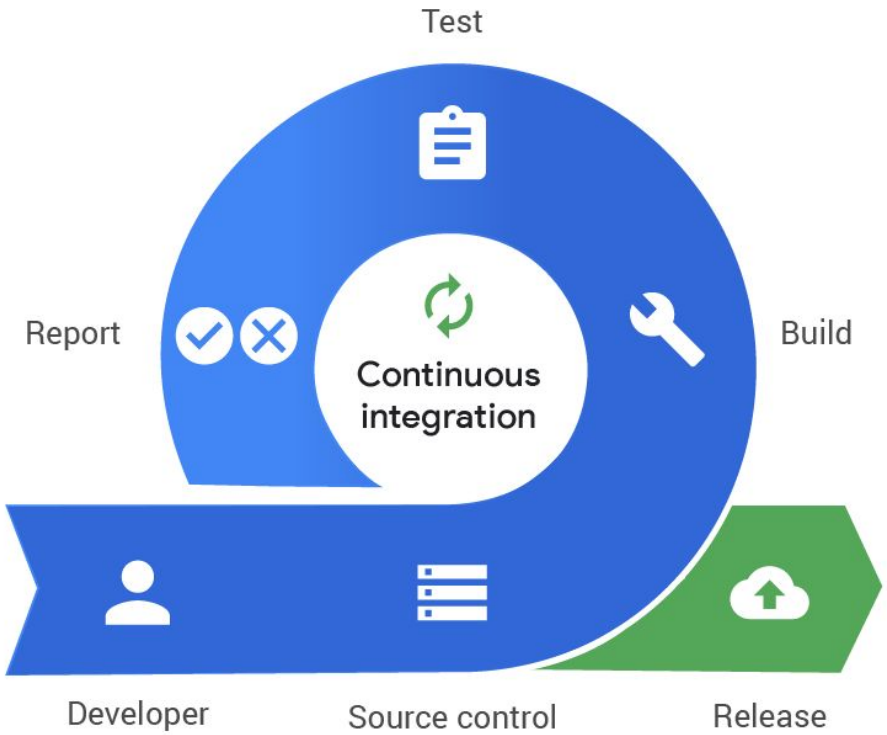


Continuous Deployment (CD)



Continuous Delivery (CD)

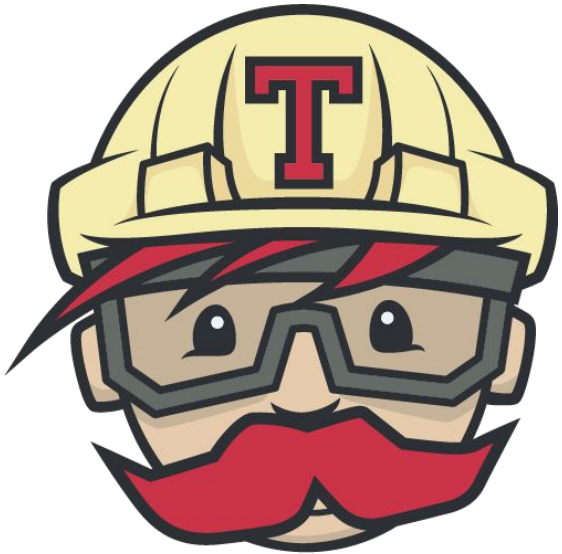
Phases of CI


Test
Report
Build
Continuous integration
Developer
Source control
Release

Tools for CI

GitHub Actions

Jenkins

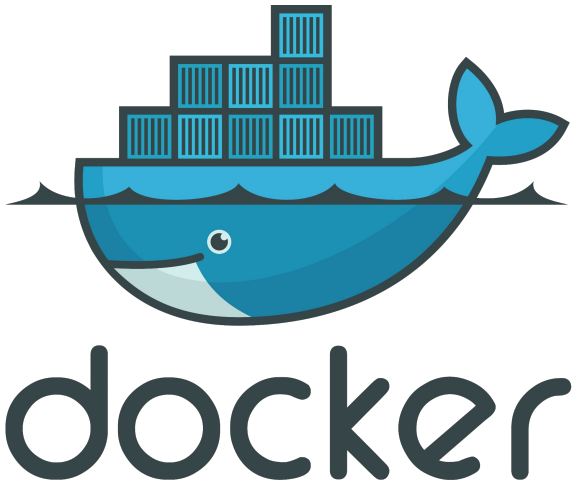


Travis CI

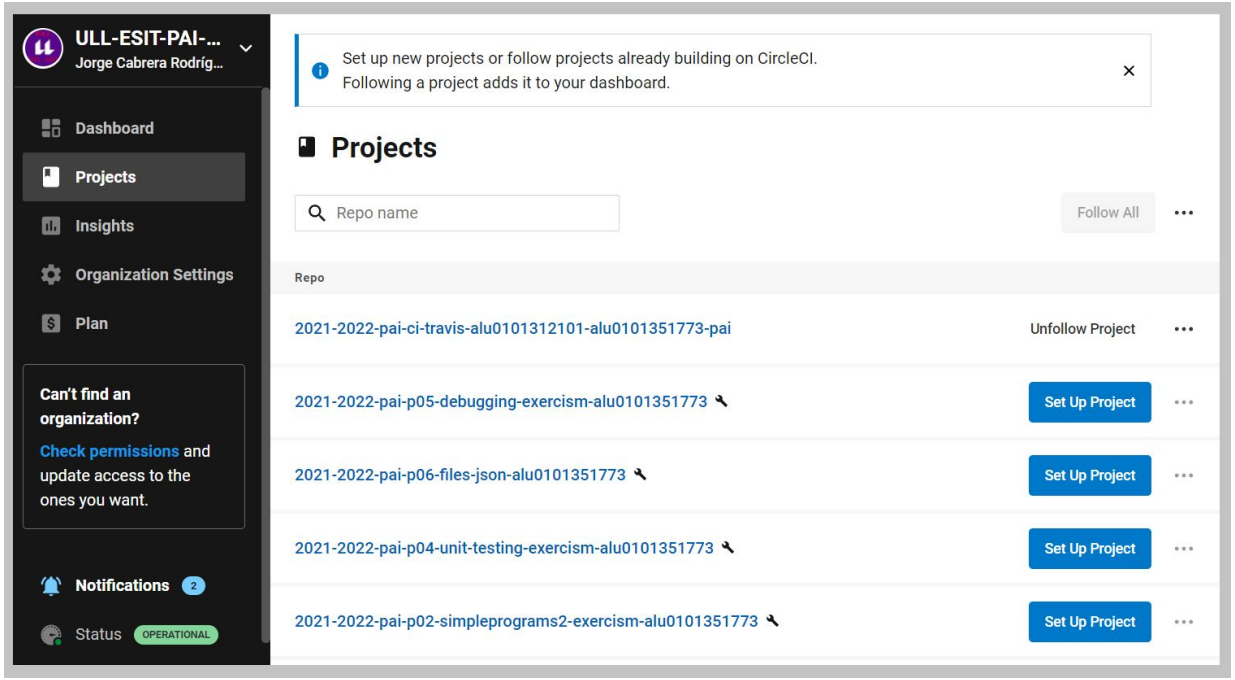
circleci

Basics of CircleCI

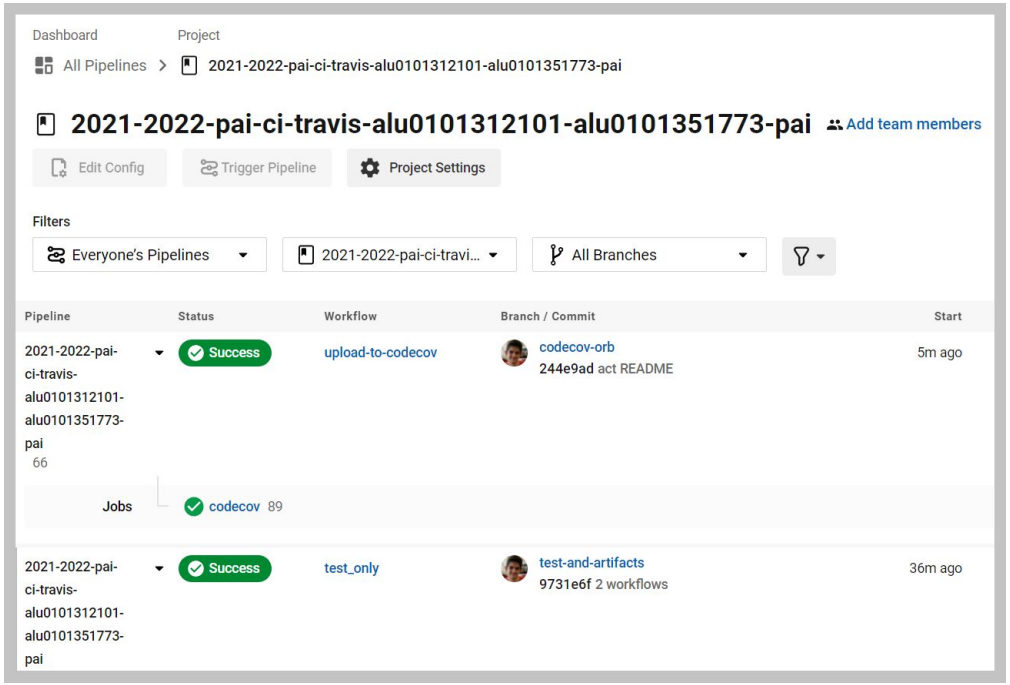
- Based on Docker
- Allow us to make automatic tasks in our project
- Easy to use
- Extremely documented
- Activated in PAI organization



Setting up a repo



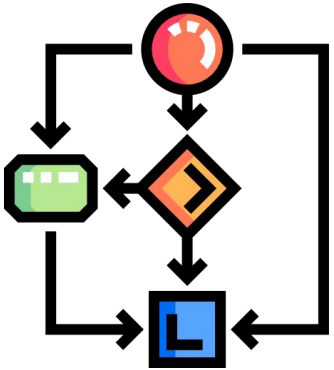
Main page of our CI



CI basic concepts

Pipelines

Workflows

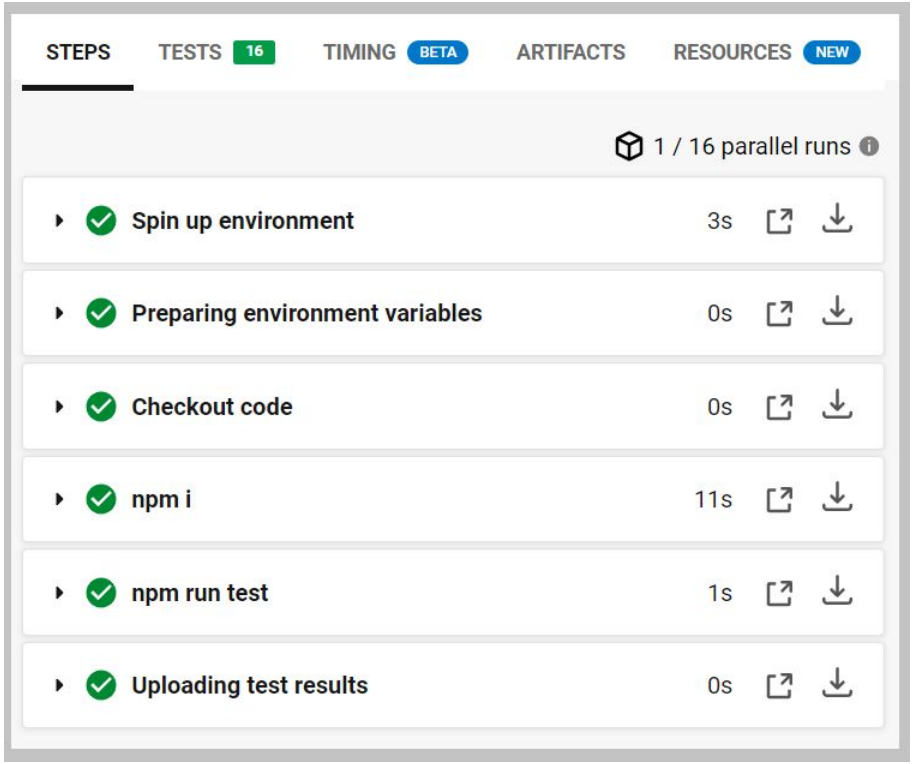


Jobs

Steps

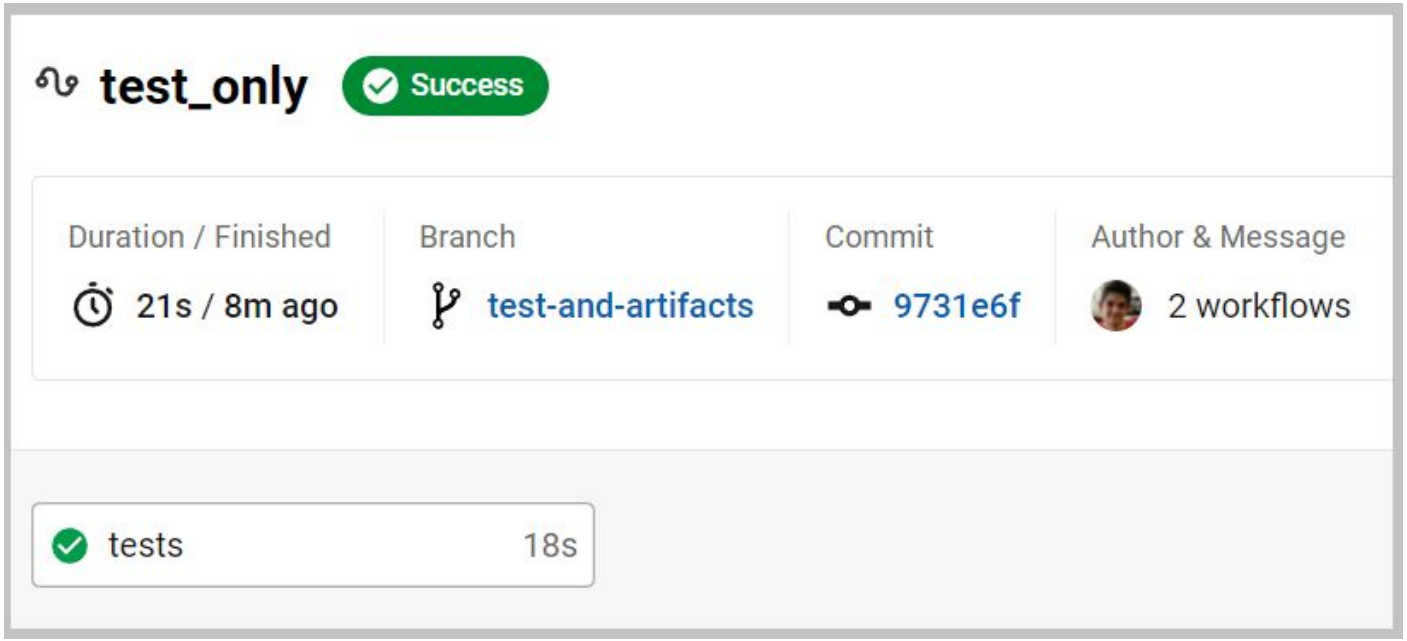
Steps

- Command of the system
- Allow us to run tasks in our virtual machine



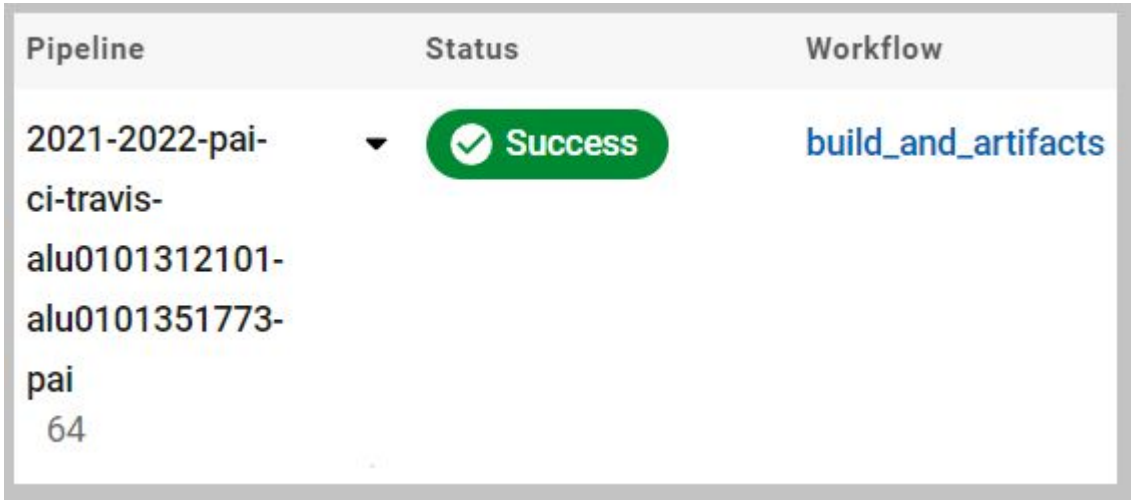
Jobs

- Set of steps



Workflows

- Structure for managing jobs
- Allow us to manage which jobs we want to be runned



Each time we run a workflow...

Very important!

All steps in the same job are executed **sequentially**.

All jobs in the same workflow are executed **parallelly**.

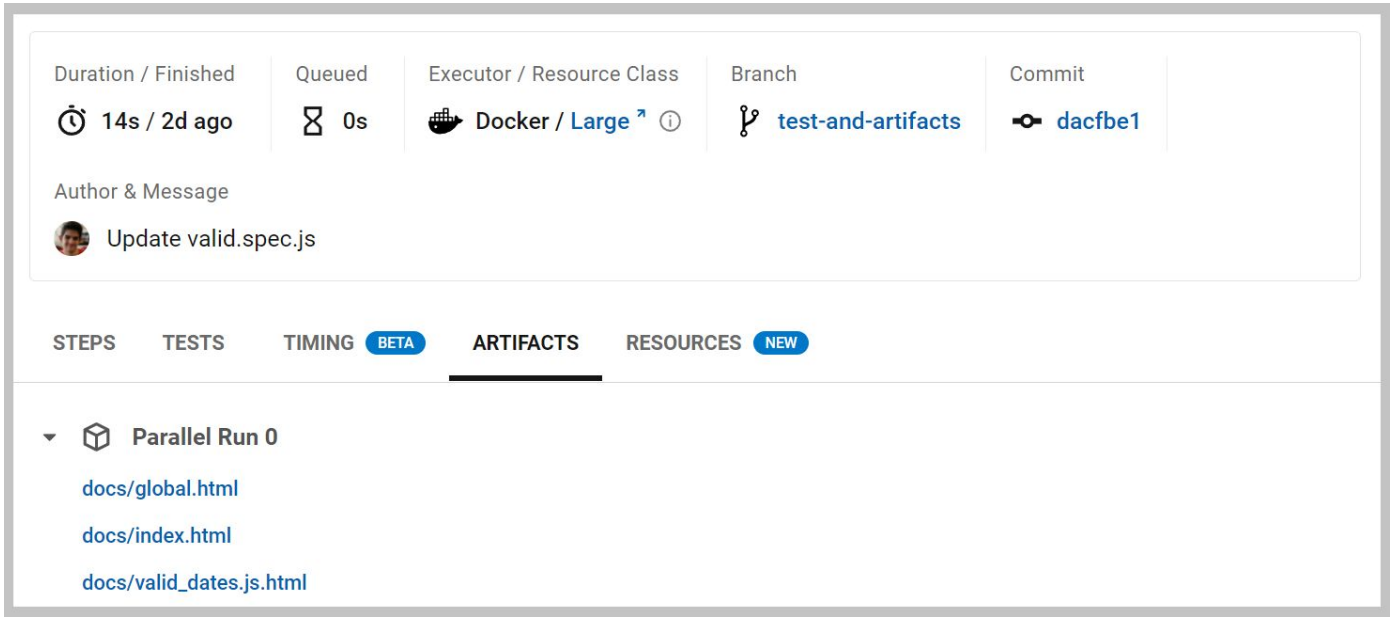
Let’s see our first automatization!

Relevant features

Test and Documentation

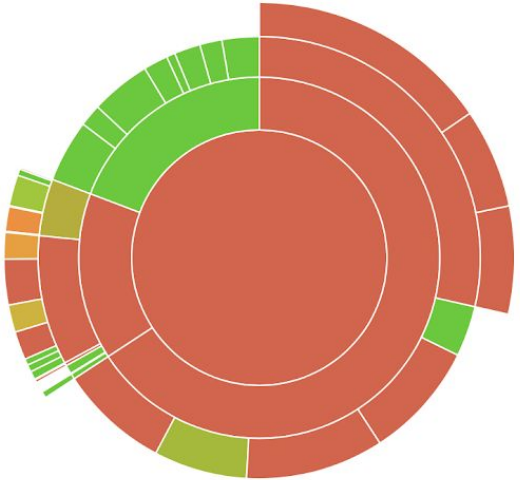
Relevant features

Artifacts



Relevant features

Code coverage



Other interesting features

Environment Variables

```
jobs:
  build-job:
    docker:
      - image: cimg/base:2020.01
    environment:
      LITTLE_CORVETTE: red
```

Context

Context: `my-context`

```
$MY_ENV_VAR = myvalue
```

```
steps: # use env var from Context
  - run: echo $MY_ENV_VAR
```

Other interesting features

Multiple Branches Execution

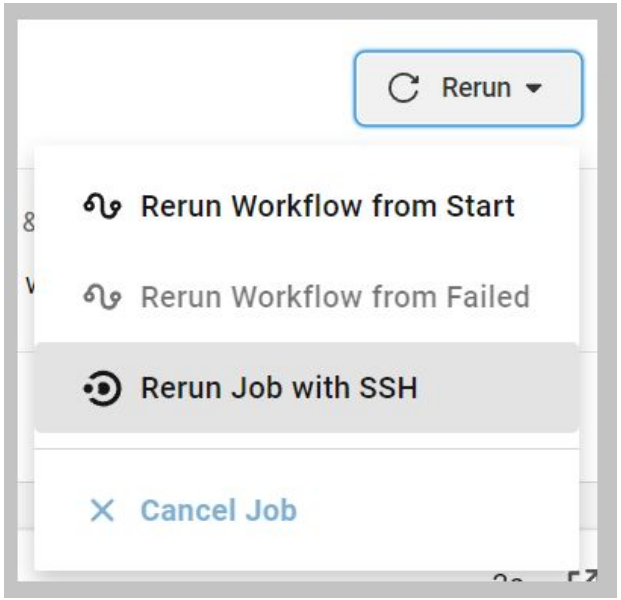
```
filters:
  branches:
    only:
      - dev
      - /user-.*/
```

Job approval

```
- hold:
    type: approval
    requires:
      - build
      - test
- deploy:
    requires:
      - hold
```

Other interesting features

SHH



Schedules

```
nightly:
  triggers:
    - schedule:
        cron: "0 0 * * *"
        filters:
```

Our personal opinion

Bibliografía

- YT Video - What is Continuous Integration?
- Wikipedia - Continuous Integration definition
- Another definition of CI and its uses
- CircleCI
- CircleCI getting started
- 14 Examples of CI tools

Bibliografía (CircleCI)

- Workflows
- Orbs
- Docker
- Store test results
- Artifacts
- Code Coverage

Questions ?

alejandro.garcia.perdomo.18@ull.edu.es

cabrera.rodriguez.12@ull.edu.es