

Exercicio 1: Probas automatizadas mediante CI/CD

Nesta tarefa non é necesario utilizar a máquina virtual de tarefas anteriores. Podes facelo dende o equipo anfitrión.

Comezaremos o exercicio creando un repositorio ne GitLab e engadindo o código da nosa aplicación.

1. Crea un novo proxecto (en GitLab chámanlle proxectos ao repositorios) de nome `iniciaist0203` con ficheiro `Readme.md`. **Realiza unha captura** da interface web co proxecto en GitLab.

The screenshot shows the GitLab web interface in a Mozilla Firefox browser. The address bar shows the URL `https://gitlab.com/d25vm/dvmt0203`. The page title is "david Vidal / dvmt0203 · GitLab". A notification banner at the top states: "Project 'dvmt0203' was successfully created." Below this, there are links to "README", "Add LICENSE", "Add CHANGELOG", "Add CONTRIBUTING", "Enable Auto DevOps", "Add Kubernetes cluster", "Set up CI/CD", "Add Wiki", and "Configure Integrations". The "Created on" date is "October 03, 2025". The "main" branch is selected for the "dvmt0203" repository. A table shows the "Initial commit" by "david Vidal" with the commit hash "90850b29". Below the table, the "README.md" file is displayed, containing the following content:

```
dvmt0203

Getting started

To make it easy for you to get started with GitLab, here's a list of recommended next steps.

Already a pro? Just edit this README.md and make it your own. Want to make it easy? Use the template at the bottom!

Add your files

☐ Create or upload files
☐ Add files using the command line or push an existing Git repository with the following command:

cd existing_repo
git remote add origin https://gitlab.com/d25vm/dvmt0203.git
```

2. Clona o repositorio.
3. Dentro do repositorio crea o seguinte ficheiro `dni_validator.py` con este contido:

```
import re

def validar_dni(dni):
    return True
```

4. Crea tamén o ficheiro `test_dni_validator.py` co seguinte contido:

```
import unittest
from dni_validator import validar_dni

class TestDniValidator(unittest.TestCase):

    def test_dni_correctos(self):
        self.assertTrue(validar_dni("12345678Z")) # Exemplo válido
        self.assertTrue(validar_dni("00000000T"))
        self.assertTrue(validar_dni("98765432M"))
        self.assertTrue(validar_dni("99999999R"))

    def test_dni_incorrectos(self):
        self.assertFalse(validar_dni("12345678A")) # Letra incorrecta
        self.assertFalse(validar_dni("1234567Z")) # Faltan números
        self.assertFalse(validar_dni("123456789Z")) # Sobran números
        self.assertFalse(validar_dni("ABCDEFGHYZ")) # Letras no número
        self.assertFalse(validar_dni("12345678")) # Sen letra
        self.assertFalse(validar_dni("")) # Baleiro

if __name__ == '__main__':
    unittest.main()
```

5. Sube o código ao repositorio en GitLab.com.

O noso repositorio conta con dous ficheiros:

- `dni_validator.py`: contén unha función que valida o DNI (de momento sen implantar)
- `test_dni_validator.py`: conta cuns test unitario para comprobar o correcto funcionamento desa función.

De seguido utilizaremos **CI/CD de GitLab para realizar o test automaticamente** en canto se suba unha nova versión a GitLab.com

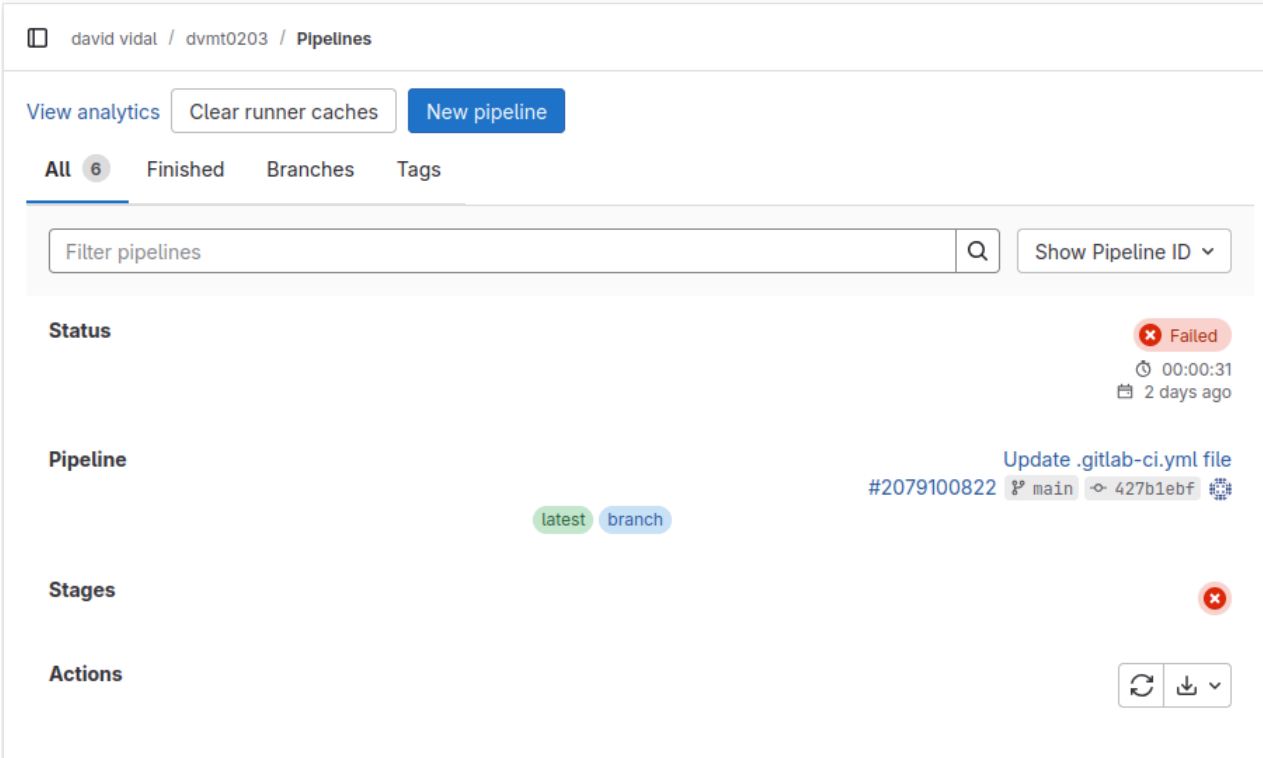
1. Crea o ficheiro `.gitlab-ci.yml`.
2. Neste ficheiro define unha única `stage` de nome `probas`.
3. Crea un novo `job` para a `stage` anterior de nome `unitarias`. Neste `job` debes de:
 - Utiliza a imaxe: `python:3.9`
 - Executar os seguintes comandos (O segundo destes comandos é o que executa as probas):

```
echo "Comezando os test..."
python -m unittest test_dni_validator.py
```

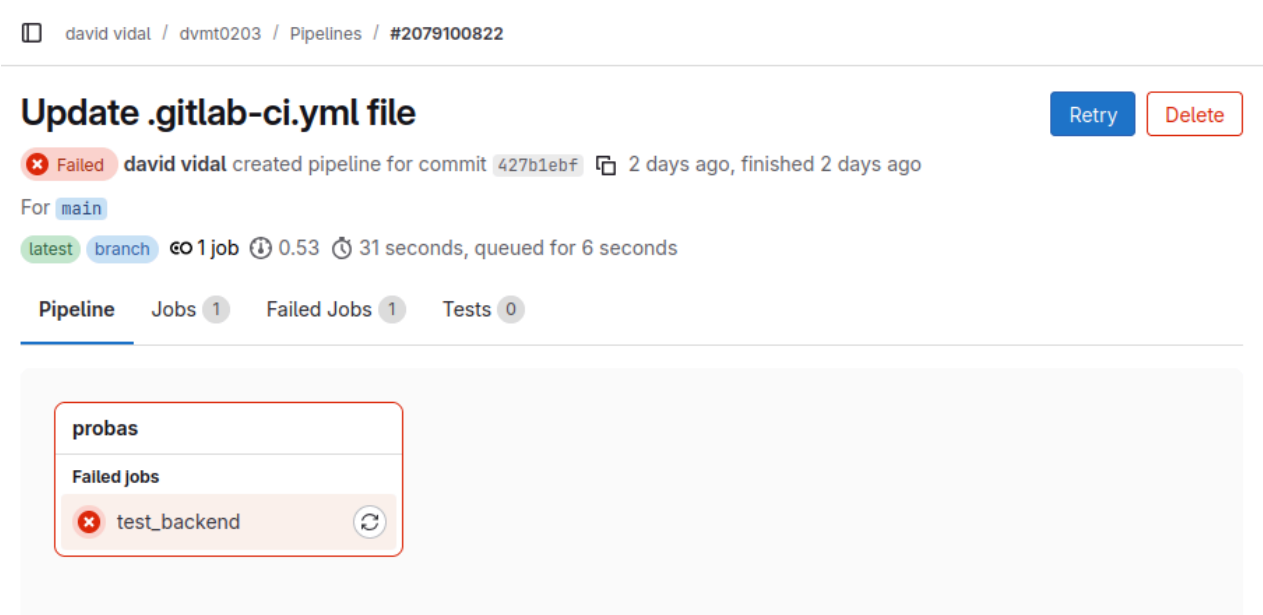
4. Fai un `commit` e un `push`. **Entrega captura** do contido do ficheiro `.gitlab-ci.yml`.

Unha vez subida a nova versión de repositorio, debería lanzarse automaticamente as probas en GitLab.com. **Vamos a ver os resultados destas probas na interface de GitLab.**

1. Abre o proxecto, e vai no menú a `Build > Pipelines`. **Realiza unha captura dos pipelines** lanzados.



2. Deberías ver que o estado é `Failed`. Preme no estado e **realiza unha captura** do contido que se mostra. Deberías ver un resumo do `pipeline` que se executou.



3. Agora selecciona a lapela `jobs`. Na táboa que se mostra, selecciona o enlace da columna `job` para ver

a execución deste. **Realiza unha captura** da execución.

test_backend

 Failed Started 2 days ago by  david vidal



```
Search visible log output [Q] [D] [P] [U] [D] [R]

1 Running with gitlab-runner 18.4.0~pre.115.gb2218bab (b2218bab)
2 on blue-1.saas-linux-small.runners-manager.gitlab.com/default j1aLDqxSn,
  system ID: s_ccdc2f364be8
3 Preparing the "docker+machine" executor 00:21
4 Using Docker executor with image python:3.11 ...
5 Using effective pull policy of [always] for container python:3.11
6 Pulling docker image python:3.11 ...
7 Using docker image sha256:a31a35e4126639ba5bf56342b284003aef1bf0e4d48707126
  b215f64d1da4a7a for python:3.11 with digest python@sha256:d64744243357bfac2
  d6fa45fb913111ec9ee513f409d505450a4b346498e85df ...
8 Preparing environment 00:06
9 Using effective pull policy of [always] for container sha256:5016f6859abd5b
  43856a106e02a446bbccad34a0584aae1f30f6e9b9874f6ada
10 Running on runner-j1aldqxn-project-75007639-concurrent-0 via runner-j1aldq
  xsn-s-l-s-amd64-1759510967-31296d9c...
11 Getting source from Git repository 00:01
12 Gitaly correlation ID: 5e2e7920290140dc8cfb2a60016f5735
13 Fetching changes with git depth set to 20...
14 Initialized empty Git repository in /builds/d25vm/dvmt0203/.git/
15 Created fresh repository.
16 Checking out 427b1ebf as detached HEAD (ref is main)...
17 Skipping Git submodules setup
18 $ git remote set-url origin "${CI_REPOSITORY_URL}" || echo 'Not a git repos
  itory; skipping'
19 Executing "step_script" stage of the job script 00:01
20 Using effective pull policy of [always] for container python:3.11
21 Using docker image sha256:a31a35e4126639ba5bf56342b284003aef1bf0e4d48707126
  b215f64d1da4a7a for python:3.11 with digest python@sha256:d64744243357bfac2
  d6fa45fb913111ec9ee513f409d505450a4b346498e85df ...
22 $ echo "Comezando os test..."
23 Comezando os test...
24 $ python -m unittest test_dni_validator.py
25 .F
26 =====
27 FAIL: test_dni_incorrectos (test_dni_validator.TestDniValidator.test_dni_in
  correctos)
28 -----
29 Traceback (most recent call last):
30   File "/builds/d25vm/dvmt0203/test_dni_validator.py", line 13, in test_dni
```

Duration: 31 seconds

Finished: 2 days ago

Queued: 0 seconds



Timeout: 1h (from project) ?

Runner: #12270807 (j1aLDqxS) 1-
blue.saas-linux-small-
amd64.runners-
manager.gitlab.com/default

Source: Push

Commit 427b1ebf 

Update .gitlab-ci.yml file

Pipeline #2079100822  Failed for ma
in 

probas

Related jobs

→  test_backend

No última captura podemos observar que a nosa función non pasou os test unitario. Lóxico xa que non está implantada. Agora poderíamos ir implantando a nosa función e facendo `push` ata que a nosa función estea correctamente implantada. Vamos xa ver como sería o resultado correcto.

1. Modifica o ficheiro `dni_validator.py` con este contido:

```
import re

def validar_dni(dni):
    """
    Valida se un DNI español é correcto en formato e letra.

    Args:
        dni (str): DNI a validar (ex: '12345678Z')

    Returns:
        bool: True se é válido, False se non.
```

```

"""
dni = dni.upper()
patron = r'^\d{8}[A-Z]$\

if not re.match(patron, dni):
    return False

letras = "TRWAGMYFPDXBNJZSQVHLCKE"
numero = int(dni[:8])
letra_correcta = letras[numero % 23]

return dni[-1] == letra_correcta

```

2. Fai un `commit` e un `push`.

3. Abre o proxecto, e vai no menú a `Build > Pipelines`. **Realiza unha captura** dos `pipelines` lanzados.

<div> All 7 Finished Branches Tags </div> <div> View analytics Clear runner caches New pipeline </div>				
<div> Filter pipelines Show Pipeline ID </div>				
Status	Pipeline	Created by	Stages	Actions
<div>Passed</div> <div>00:00:33</div> <div>44 seconds ago</div>	<div>dnvalidatorfix</div> <div>#2082716405</div> <div>fdacc48</div> <div>latest branch</div>		<div></div>	<div>Download</div>
<div>Failed</div> <div>00:00:31</div> <div>2 days ago</div>	<div>Update .gitlab-ci.yml file</div> <div>#2079100822</div> <div>427b1ebf</div> <div>branch</div>		<div></div>	<div>Refresh</div> <div>Download</div>

4. Deberías ver que o estado é `Passed`. Preme no estado e **realiza unha captura** do contido que se mostra. Deberías ver un resumo do `pipeline` que se executou.

david vidal / dvmt0203 / Pipelines / #2082716405

dnvalidatorfix

Delete

Passed

 david vidal created pipeline for commit fdacc48 3 minutes ago, finished 2 minutes ago

For main

latest branch 1 job 0.55 33 seconds, queued for 1 seconds

Pipeline Jobs 1 Tests 0


probas

test_backend

5. Agora selecciona a lapela `jobs`. Na táboa que se mostra, selecciona o enlace da columna `job` para ver

a execución deste. **Realiza unha captura** da execución.

📄 david vidal / dvmt0203 / Jobs / #11612839199

✓ Passed Started 3 minutes ago by  david vidal



Search visible log output



```
1 Running with gitlab-runner 18.4.0~pre.115.gb2218bab (b2218bab)
2 on blue-4.saas-linux-small-amd64.runners-manager.gitlab.com/default J2nyw
w-sK, system ID: s_cf1798852952
3 Preparing the "docker+machine" executor 00:22
4 Using Docker executor with image python:3.11 ...
5 Using effective pull policy of [always] for container python:3.11
6 Pulling docker image python:3.11 ...
7 Using docker image sha256:a31a35e4126639ba5bf56342b284003aef1bf0e4d48707126
b215f64d1da4a7a for python:3.11 with digest python@sha256:d64744243357bfac2
d6fa45fb913111ec9ee513f409d505450a4b346498e85df ...
8 Preparing environment 00:05
9 Using effective pull policy of [always] for container sha256:6fca81730ff767
a6e624341a4f4c64569194e1bf92c6dfe105950346f363d861
10 Running on runner-j2nyww-sk-project-75007639-concurrent-0 via runner-j2nyw
w-sk-s-l-s-amd64-1759731279-6b40cb64...
11 Getting source from Git repository 00:02
12 Gitaly correlation ID: 5845e36781524cd49ac39278f21a1fbc
13 Fetching changes with git depth set to 20...
14 Initialized empty Git repository in /builds/d25vm/dvmt0203/.git/
15 Created fresh repository.
16 Checking out fdacc48 as detached HEAD (ref is main)...
17 Skipping Git submodules setup
18 $ git remote set-url origin "${CI_REPOSITORY_URL}" || echo 'Not a git repos
itory; skipping'
19 Executing "step_script" stage of the job script 00:01
20 Using effective pull policy of [always] for container python:3.11
21 Using docker image sha256:a31a35e4126639ba5bf56342b284003aef1bf0e4d48707126
b215f64d1da4a7a for python:3.11 with digest python@sha256:d64744243357bfac2
d6fa45fb913111ec9ee513f409d505450a4b346498e85df ...
22 $ echo "Comezando os test..."
23 Comezando os test...
24 $ python -m unittest test_dni_validator.py
25 ..
26 -----
27 Ran 2 tests in 0.000s
28 OK
29 Cleaning up project directory and file based variables 00:01
30 Job succeeded
```

Duration: 33 seconds


Finished: 3 minutes ago

Queued: 0 seconds


Timeout: 1h (from project) ?

Runner: #12270837 (J2nyww-s) 4-
blue.saas-linux-small-
amd64.runners-
manager.gitlab.com/default

Source: Push

Commit fdacc48 

dnivalidatorfix

Pipeline #2082716405 ✓ Passed for m
ain 

probas

Related jobs

→ ✓ test_backend

Exercicio 2: Build automatizado mediante CI/CD

Neste momento sabemos que a nosa función funciona correctamente porque pasa tódalas probas unitarias. Poderíamos utilizar esta función nun programa principal, no que se recibira o dni por argumento e se indicara se é correcto ou non.

1. Crea no repositorio o ficheiro `main.py` co seguinte contido:

```
import sys
from dni_validator import validar_dni

def main():
    if len(sys.argv) != 2:
        print("Uso: python main.py <DNI>")
        sys.exit(1)

    dni = sys.argv[1]
```

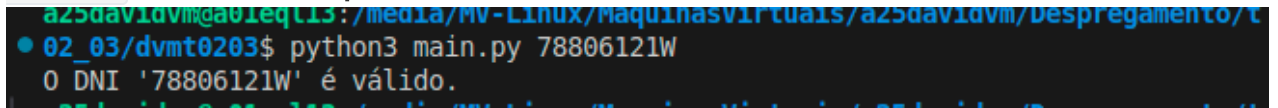
```

def validar_dni(dni):
    print(f"O DNI '{dni}' é válido.")
else:
    print(f"O DNI '{dni}' NON é válido.")

if __name__ == "__main__":
    main()

```

2. Executa o seguinte comando para ver se funciona correctamente o programa: `python3 main.py <o teu DNI>`. **Realiza unha captura** da execución do comando.



3. Sube o repositorio ao GitLab.

Agora imos **automatizar o empaquetado do noso programa**, pero só se realizará se pasa as probas correctamente. Para iso engadiremos unha nova `stage`.

1. Engade unha nova `stage` que se execute só se o resultado das probas é satisfactorio. O nome da `stage` será `build`.
2. Crea un novo `job` para a `stage` anterior de nome `empaquetado`. Neste `job` debes de:
 - o Utiliza a imaxe: `python:3.9`
 - o Executar os seguintes comandos (O seguindo destes comandos e o que executa as probas):

```

pip install pyinstaller # Instala o paquete que nos permite empaquetar o
paquete
pyinstaller --onefile main.py # Realiza o empaquetado. O executable crease no
directorio dist co nome main

```

- o Engade o seguinte dentro do `job`. Isto permitiranos acceder ao executable dende a interface e poder descargalo.

```

artifacts:
  paths:
    - dist/main
  expire_in: 1 week

```

3. Sube o repositorio a GitLab. **Entrega captura** do contido do ficheiro `.gitlab-ci.yml`.

main / .gitlab-ci.yml

.gitlab-ci.yml

Find file

Blame

Edit



taberror2

david vidal authored 1 minute ago



f707b4a7



History



.gitlab-ci.yml 504 B



```
1
2 stages:
3   - probas
4   - build
5
6 test_backend:
7   stage: probas
8   image: python:3.11
9   script:
10    - echo "Comezando os test..."
11    - python -m unittest test_dni_validator.py
12
13 empaquetado:
14   stage: build
15   image: python:3.9
16   artifacts:
17     paths:
18       - dist/main
19     expire_in: 1 week
20   script:
21     - pip install pyinstaller # Instala o paquete que nos permite empaquetar o paquete
22     - pyinstaller --onefile main.py # Realiza o empaquetado. O executable crease no directorio dist co nome main
23
```

4. Abre o proxecto, e vai no menú a **Build > Pipelines**. **Realiza unha captura** dos **pipelines**

lanzados.

🏠 david vidal / dvmt0203 / Pipelines

All 12

Finished

Branches

Tags

































View analytics

Clear runner caches

New pipeline

Filter pipelines


Show Pipeline ID

Status	Pipeline	Created by	Stages	Actions
<div>✔ Passed</div> <div>⌚ 00:01:18</div> <div>📅 45 seconds ago</div>	<div>taberror2</div> <div>#2082748453 </div> <div>🔗 f707b4a7 </div> <div>latest  branch </div>		<div>✔✔</div>	<div>📄</div>
<div>✖ Failed</div> <div>📅 5 minutes ago</div>	<div>taberror</div> <div>#2082743359 </div> <div>🔗 5483f4a7 </div> <div>error  branch </div>			<div>📄</div>
<div>✖ Failed</div> <div>📅 6 minutes ago</div>	<div>nomeben</div> <div>#2082741653 </div> <div>🔗 7c99bc4c </div> <div>error  branch </div>			<div>📄</div>
<div>✖ Failed</div> <div>📅 8 minutes ago</div>	<div>autobuild</div> <div>#2082738388 </div> <div>🔗 a4bc0db7 </div> <div>error  branch </div>			<div>📄</div>
<div>✔ Passed</div> <div>⌚ 00:00:31</div> <div>📅 15 minutes ago</div>	<div>main</div> <div>#2082727641 </div> <div>🔗 4c7a4bab </div> <div>branch </div>		<div>✔</div>	<div>📄</div>
<div>✔ Passed</div> <div>⌚ 00:00:33</div> <div>📅 20 minutes ago</div>	<div>dnivalidatorfix</div> <div>#2082716405 </div> <div>🔗 fdaccc48 </div> <div>branch </div>		<div>✔</div>	<div>📄</div>
<div>✖ Failed</div> <div>⌚ 00:00:31</div> <div>📅 2 days ago</div>	<div>Update .gitlab-ci.yml file</div> <div>#2079100822 </div> <div>🔗 427b1ebf </div> <div>branch </div>		<div>✖</div>	<div>🔄📄</div>

5. Deberías ver que o estado é `Passed`. Na columna etapas, deberías ver que hai dous círculos en verde que identifican as dúas `stages` que temos. Preme no segundo e despois no `job` empaquetado.

Realiza **captura** do contido da web que se mostra.

empaquetado

✓ Passed Started 2 minutes ago by  david vidal



Search visible log output

```
1 Running with gitlab-runner 18.4.0~pre.115.gb2218bab (b2218bab)
2 on blue-3.saas-linux-small-amd64.runners-manager.gitlab.com/default zxwgk
  jAPH, system ID: s_d5d3abbdfdf0a
3 Preparing the "docker+machine" executor 00:21
4 Using Docker executor with image python:3.9 ...
5 Using effective pull policy of [always] for container python:3.9
6 Pulling docker image python:3.9 ...
7 Using docker image sha256:960cbacde4487a3809a1c6487661f695870b606f547b2addb
  5bc578498ac7828 for python:3.9 with digest python@sha256:1ed76782d6e927c60c
  7e631a3642cee0f4fe786401c73645e3bb646f36137234 ...
8 Preparing environment 00:06
9 Using effective pull policy of [always] for container sha256:8ca514a37f8e61
  c3b4cb8464be8c0569bb79e0a37a0fd71dffe9c286b8c70f47
10 Running on runner-zxwgkjaph-project-75007639-concurrent-0 via runner-zxwgkj
  aph-s-l-s-amd64-1759735460-eddf6f78...
11 Getting source from Git repository 00:01
12 Gitaly correlation ID: ec0c669deb9548b9ac799c3bbb27710f
13 Fetching changes with git depth set to 20...
14 Initialized empty Git repository in /builds/d25vm/dvmt0203/.git/
15 Created fresh repository.
16 Checking out f707b4a7 as detached HEAD (ref is main)...
17 Skipping Git submodules setup
18 $ git remote set-url origin "${CI_REPOSITORY_URL}" || echo 'Not a git repos
  itory; skipping'
19 Executing "step_script" stage of the job script 00:13
20 Using effective pull policy of [always] for container python:3.9
21 Using docker image sha256:960cbacde4487a3809a1c6487661f695870b606f547b2addb
  5bc578498ac7828 for python:3.9 with digest python@sha256:1ed76782d6e927c60c
  7e631a3642cee0f4fe786401c73645e3bb646f36137234 ...
22 $ pip install pyinstaller
23 Collecting pyinstaller
24   Downloading pyinstaller-6.16.0-py3-none-manylinux2014_x86_64.whl (733 kB)
25   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 733.8/733.8 kB 10.2 MB/s eta
  0:00:00
26 Collecting pyinstaller-hooks-contrib>=2025.8
27   Downloading pyinstaller_hooks_contrib-2025.9-py3-none-any.whl (444 kB)
28   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 444.3/444.3 kB 40.5 MB/s eta
  0:00:00
29 Requirement already satisfied: setuptools>=42.0.0 in /usr/local/lib/python
```

Duration: 46 seconds

Finished: 1 minute ago

Queued: 0 seconds

Timeout: 1h (from project) ?

Runner: #12270835 (zxwgkJP) 3-
blue.saas-linux-small-
amd64.runners-
manager.gitlab.com/default

Source: Push

Job artifacts ?

These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.

Keep

Download

Browse

Commit f707b4a7

taberror2

Pipeline #2082748453 ✓ Passed for
main

build

Related jobs


→ ✓ empaquetado

6. Na parte dereita podes ver un apartado que pon **Artefactos e jobs**. Dende aí podes premer en **Descargar**. Preme e baixase o noso programa empaquetado. Descomprime o **zip**.

7. No terminal do equipo, vaite ao directorio onde descargaches o programa anterior. Executa o seguinte

comando: `./main <o teu DNI>`. Realiza **captura** da execução deste comando.

empacquetado

✓ Passed Started 2 minutes ago by  david vidal



Search visible log output 🔍 📄 ↺ ⬆️ ⬇️ ↗️

```
1 Running with gitlab-runner 18.4.0~pre.115.gb2218bab (b2218bab)
2   on blue-3.saas-linux-small-amd64.runners-manager.gitlab.com/default zwxgk
  jAPH, system ID: s_d5d3abbdfd0a
3   ✓ Preparing the "docker+machine" executor 00:21
4   Using Docker executor with image python:3.9 ...
5   Using effective pull policy of [always] for container python:3.9
6   Pulling docker image python:3.9 ...
7   Using docker image sha256:960cbacde4487a3809a1c6487661f695870b606f547b2addb
  5bc578498ac7828 for python:3.9 with digest python@sha256:1ed76782d6e927c60c
  7e631a3642cee0f4fe786401c73645e3bb646f36137234 ...
8   ✓ Preparing environment 00:06
9   Using effective pull policy of [always] for container sha256:8ca514a37f8e61
  c3b4cb8464be8c0569bb79e0a37a0fd71dffe9c286b8c70f47
10  Running on runner-zwxgkjaph-project-75007639-concurrent-0 via runner-zwxgkj
  aph-s-l-s-amd64-1759735460-eddf6f78...
11  ✓ Getting source from Git repository 00:01
12  Gitaly correlation ID: ec0c669deb9548b9ac799c3bbb27710f
13  Fetching changes with git depth set to 20...
14  Initialized empty Git repository in /builds/d25vm/dvmt0203/.git/
15  Created fresh repository.
16  Checking out f707b4a7 as detached HEAD (ref is main)...
17  Skipping Git submodules setup
18  $ git remote set-url origin "${CI_REPOSITORY_URL}" || echo 'Not a git repos
  itory; skipping'
19  ✓ Executing "step_script" stage of the job script 00:13
20  Using effective pull policy of [always] for container python:3.9
21  Using docker image sha256:960cbacde4487a3809a1c6487661f695870b606f547b2addb
  5bc578498ac7828 for python:3.9 with digest python@sha256:1ed76782d6e927c60c
  7e631a3642cee0f4fe786401c73645e3bb646f36137234 ...
22  $ pip install pyinstaller
23  Collecting pyinstaller
24    Downloading pyinstaller-6.16.0-py3-none-manylinux2014_x86_64.whl (733 kB)
25    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 733.8/733.8 kB 10.2 MB/s eta
  0:00:00
26  Collecting pyinstaller-hooks-contrib>=2025.8
27    Downloading pyinstaller_hooks_contrib-2025.9-py3-none-any.whl (444 kB)
28    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 444.3/444.3 kB 40.5 MB/s eta
  0:00:00
29  Requirement already satisfied: setuptools>=42.0.0 in /usr/local/lib/python
```

Duration: 46 seconds
Finished: 1 minute ago
Queued: 0 seconds
Timeout: 1h (from project) ⓘ
Runner: #12270835 (zwxgkjAP) 3-blue.saas-linux-small-amd64.runners-manager.gitlab.com/default
Source: Push

Job artifacts ⓘ
These artifacts are the latest. They will not be deleted (even if expired) until newer artifacts are available.

Keep Download Browse

Commit f707b4a7 📄
taberror2

Pipeline #2082748453 ✓ Passed for main 📄

build ▾

Related jobs
→ ✓ empacquetado