

<<<<<< HEAD

Segundo Parcial Jenkins

Autor: Jose Luis Sacanamboy Rozo**Código:** 13103010**Repositorio:** <https://github.com/jose0luis41/testproject>

Desarrollo Parcial

Se instala el Jenkins con sus dependencias.

Iniciamos el servicio de Jenkins y habilitamos su puerto

Segundo Parcial Operativos

Nombre: Jose Luis Sacanamboy**Código:** 13103010

Desarrollo

Este parcial consta de la realización de pruebas a través de la herramienta Jenkins, a los servicios que se realizaron en el anterior parcial.

Configuración Jenkins

Primero se configura la fecha del servidor de Jenkins.

```
yum install ntp ntpdate ntp-doc
chkconfig ntpd on
ntpdate pool.ntp.org
/etc/init.d/ntpd start
```

Una vez configurada la hora del servidor proseguimos a instalar el Jenkins.

| | | | | | 171ab3e5cdf90c68e327e7af76a126edf2d1b973

creamos el usuario y el ambiente virtual para Jenkins

<<<<<< HEAD

Se instala las herramientas para el parcial y exportar el archivo de requerimientos.

Se ha instalado el Jenkins correctamente. Proseguimos activar el servicio web de Jenkins y el puerto 8080 por el cual el

Jenkins sube.

```
chkconfig jenkins on
service jenkins start
iptables -I INPUT 5 -p tcp -m state --state NEW -m tcp --dport 8080 -j ACCEPT
service iptables save
```

Una vez instalado Jenkins continuamos a crear el ambiente virtual con el que correrá Jenkins.

```
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
pip install virtualenv

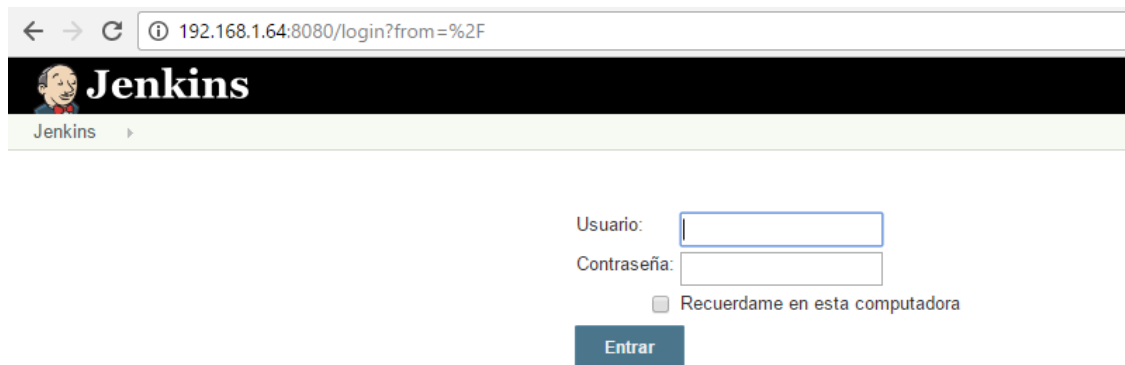
mkdir /var/lib/jenkins/.virtualenvs
cd /var/lib/jenkins/.virtualenvs
virtualenv testproject
. testproject/bin/activate
```

Activado el ambiente virtual, se sigue la instalación de las dependencias necesarias para la realización de las pruebas.


```
pip install xmlrunner
pip install unittest2
pip install pytest
pip install pytest-cov
pip install flask

pip freeze > requirements.txt
```

Finalizado. Ahora se abre el Jenkins por medio de la URL <http://192.168.1.64:8080/>, esta es respecto a la dirección de la máquina utilizada.



← → ↻ ⓘ 192.168.1.64:8080/login?from=%2F

 **Jenkins**

Jenkins ▶

Usuario:

Contraseña:


☐ Recuerdame en esta computadora

Entrar

Se procede a ingresar al Jenkins. Esto se debe a que ya se había ingresado. Por lo tanto se procede a pedir la llave desde la máquina virtual.

```
nano /var/lib/jenkins/secret.key
```

← → ↻ 192.168.1.64:8080/login?from=%2F



Jenkins

Jenkins ▶

Usuario:


Contraseña:

☐ Recuerdame en esta computadora

Entrar

Ahora se procede a configurar el Jenkins, enlazandolo con el repositorio de GitHub, el cual ya contiene la configuración de los archivos de prueba.

← → ↻ 192.168.1.64:8080/job/testproject/configure



Jenkins

Jenkins ▶ testproject ▶

General Configurar el origen del código fuente Disparadores de ejecuciones Entorno de ejecución Ejecutar

Acciones para ejecutar después.

Proyecto nombre

Descripción

[Plain text] [Visualizar](#)

☐ Desechar ejecuciones antiguas


☐ Esta ejecución debe parametrizarse

☒ GitHub project

Project url

Avanzado...

← → ↻ 192.168.1.64:8080/job/testproject/configure



Jenkins

Jenkins ▶ testproject ▶

General **Configurar el origen del código fuente** Disparadores de ejecuciones Entorno de ejecución Ejecutar

Acciones para ejecutar después.

☐ Lanzar ejecuciones concurrentes en caso de ser necesario

Configurar el origen del código fuente

☐ Ninguno

☒ Git

Repositories

Repository URL

Credentials [Add](#)

Avanzado...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

← → ↻ 192.168.1.64:8080/job/testproject/configure

Jenkins ▾ testproject ▸

General Configurar el origen del código fuente **Disparadores de ejecuciones** Entorno de ejecución Ejecutar

Acciones para ejecutar después.

Subversion ⓘ

Disparadores de ejecuciones

- ☐ Build after other projects are built ⓘ
- ☐ Build when a change is pushed to GitHub ⓘ
- ☐ Consultar repositorio (SCM) ⓘ
- ☒ Ejecutar periódicamente ⓘ

Programador

Would last have run at martes 8 de noviembre de 2016 22H01' COT; would next run at martes 8 de noviembre de 2016 22H06' COT.

Entorno de ejecución

- ☐ Delete workspace before build starts
- ☐ Abortar la ejecución si se atasca

← → ↻ 192.168.1.64:8080/job/testproject/configure

Jenkins ▾ testproject ▸

General Configurar el origen del código fuente Disparadores de ejecuciones **Entorno de ejecución** Ejecutar

Acciones para ejecutar después.

- ☐ Add timestamps to the Console Output ⓘ
- ☐ Use secret text(s) or file(s) ⓘ

Ejecutar

Ejecutar línea de comandos (shell) ⓘ

Comando

Visualizar [la lista de variables de entorno disponibles](#)

Avanzado...

Añadir un nuevo paso ▾

← → ↻ 192.168.1.64:8080/job/testproject/configure

Jenkins ▾ testproject ▸

General Configurar el origen del código fuente Disparadores de ejecuciones Entorno de ejecución **Ejecutar**

Acciones para ejecutar después.

Publicar los resultados de tests JUnit ⓘ

Ficheros XML con los informes de tests

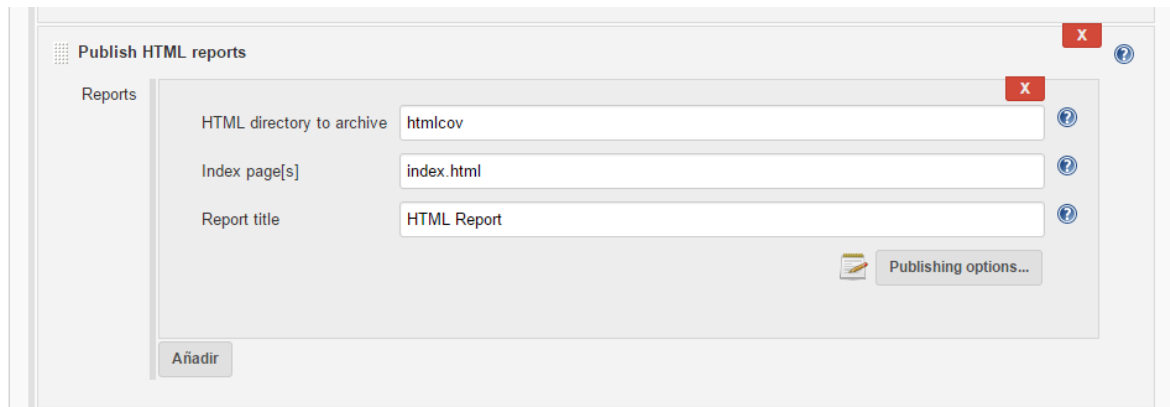
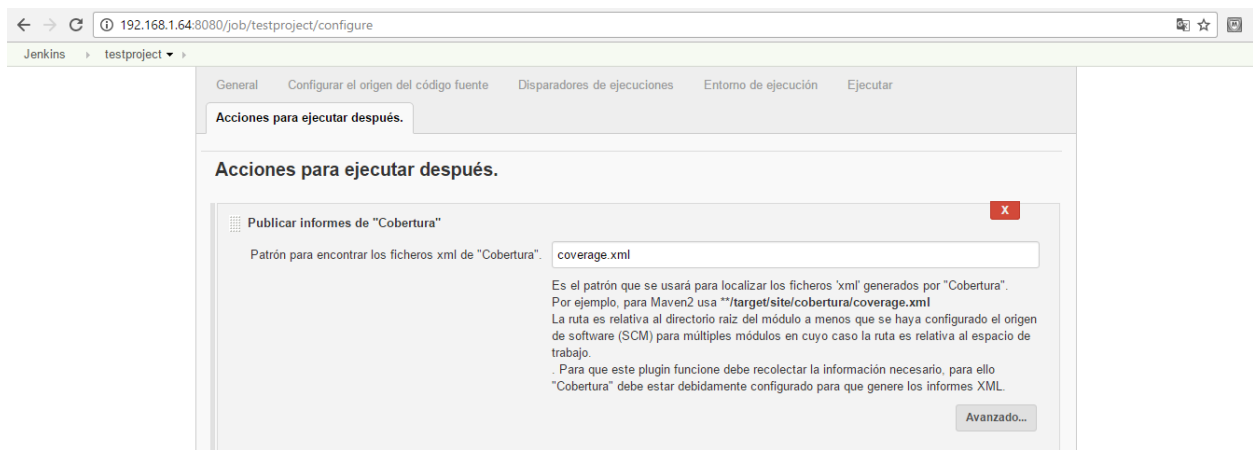
El atributo '@includes' de la etiqueta 'fileset' especifica dónde están los ficheros XML generados, por ejemplo: 'myproject/target/test-reports/*.xml'. El directorio base para la etiqueta 'fileset' es el directorio raíz del proyecto

- ☐ Guardar la salida estándar y de error aunque sea muy larga. ⓘ

Health report amplification factor

1% failing tests scores as 99% health. 5% failing tests scores as 95% health ⓘ

Allow empty results ☐ Do not fail the build on empty test results ⓘ



Codificación de las pruebas

A continuación se encuentra el código fuente de los servicios y las pruebas.

Para los servicios se expone en un puerto diferente al configurado para el Jenkins.

files.py

```

from flask import Flask, request, abort
import os, json

from files_commands import get_files, remove_all_files, get_files_recent

app = Flask(__name__)
api_url = '/v1.0'

@app.route(api_url + '/files', methods=['POST'])
def create_file():
    content = request.get_json(silent=True)
    filename = content['filename']
    content = content['content']
    os.chdir('files_created')
    result = open(filename+'.txt', 'a')
    result.write(content+'\n')
    result.close()
    os.chdir('..')
    return "CREATED", 201

@app.route(api_url + '/files', methods=['GET'])
def read_file():
    list = {}
    list["files"] = get_files()
    return json.dumps(list), 200

@app.route(api_url + '/files', methods=['PUT'])
def update_file():
    return "NOT FOUND", 404

@app.route(api_url + '/files', methods=['DELETE'])
def delete_file():
    error = False
    error = remove_all_files()
    if error:
        return 'REMOVE ALL FILES', 200
    else:
        return 'ERROR', 400

@app.route(api_url + '/files/recently_created', methods=['POST'])
def create_file_recent():
    return "NOT FOUND", 404

@app.route(api_url + '/files/recently_created', methods=['GET'])
def read_file_recent():
    list = {}
    list["files"] = get_files_recent()
    return json.dumps(list), 200

@app.route(api_url + '/files/recently_created', methods=['PUT'])
def update_file_recent():
    return "NOT FOUND", 404

@app.route(api_url + '/files/recently_created', methods=['DELETE'])
def delete_file_recent():
    return "NOT FOUND", 404

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=9090, debug=True)

```

files_commands.py

```

from subprocess import Popen, PIPE
import os

def get_files():
    list = Popen(('ls', 'files_created'), stdout=PIPE, stderr=PIPE).communicate()[0].split('\n')
    return filter(None, list)

def remove_all_files():
    execute = 'find /home/filesystem_user -type f -maxdepth 1 -not -name ".*" -exec rm {} \;'
    process = Popen(execute, universal_newlines=True, stdout=PIPE, stderr=PIPE, shell=True).communicate()
    return True

def get_files_recent():
    list = Popen(('ls', 'files_created', '-t'), stdout=PIPE, stderr=PIPE).communicate()[0].split('\n')
    return filter(None, list)

```

Pruebas

test_parcial

```

import pytest
import json, files

@pytest.fixture
def client(request):
    client = files.app.test_client()
    return client

@pytest.mark.order1
def test_Uno(client):
    testUno = {'filename': 'fileUno', 'content': 'ContenidoUno'}
    testDos = {'filename': 'fileDos', 'content': 'ContenidoDos'}
    ejecucionUno = client.post('/v1.0/files', data = json.dumps(testUno), content_type='application/json')
    ejecucionDos = client.post('/v1.0/files', data = json.dumps(testDos), content_type='application/json')
    assert ejecucionUno.status == '201 CREATED'
    assert ejecucionDos.status == '201 CREATED'
    ejecucionTres = client.get('/v1.0/files', follow_redirects=True)
    assert "fileUno" in ejecucionTres.data
    assert "fileDos" in ejecucionTres.data

@pytest.mark.order2
def test_Dos(client):
    testUno = {'filename': '', 'content': 'ContenidoUno'}
    testDos = {'filename': 'fileUno', 'content': ''}
    ejecucionUno = client.post('/v1.0/files', data = json.dumps(testUno), content_type='application/json')
    ejecucionDos = client.post('/v1.0/files', data = json.dumps(testDos), content_type='application/json')
    assert ejecucionUno.status == '400 BAD REQUEST'
    assert ejecucionDos.status == '400 BAD REQUEST'

@pytest.mark.order3
def test_Tres(client):
    ejecucion = client.get('/v1.0/files', follow_redirects=True)
    assert "fileUno" in ejecucion.data
    assert "fileDos" in ejecucion.data

@pytest.mark.order4
def test_Cuatro(client):
    ejecucion = client.get('/v1.0/files/recently_created', follow_redirects=True)
    assert "fileDos" in ejecucion.data
    assert "fileUno" in ejecucion.data

@pytest.mark.order5
def test_Cinco(client):
    client.delete('/v1.0/files', follow_redirects=True)
    ejecucion = client.get('/v1.0/files', follow_redirects=True)
    assert "fileUno" not in ejecucion.data
    assert "fileDos" not in ejecucion.data

```

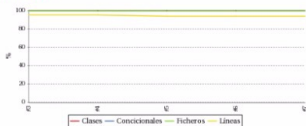
Ahora se prueban los servicios utilizando Jenkins.

Se ingresa al archivo de cobertura.

Cobertura de código

[Cobertura Coverage Report >](#)

Tendencia



Paquete (resumen)

Nombre	Ficheros	Clases	Líneas	Condicionales
-	100% 3/3	100% 3/3	94% 112/119	100% 0/0

Fichero (detalle)

Nombre	Clases	Líneas	Condicionales
files.py	100% 1/1	83% 36/49	-
files_commands.py	100% 1/1	100% 23/23	-
test_files.py	100% 1/1	100% 54/54	-

171ab3e5cdf90c68e327e7af76a126edf2d1b973