

<b>1. MÉTODO SUMA CON PYTHON Y PRUEBAS UNITARIAS CON PYTEST .....</b>	<b>2</b>
PASO 1.1: Crear archivo calc.py .....	2
PASO 1.2: Crear archivo de pruebas unitarias test.py .....	2
PASO 1.3: Requerimientos para realizar pruebas unitarias en Ubuntu .....	3
PASO 1.4: Ejecucion de pruebas unitarias con pytest .....	4
<b>2. CREAR REPOSITORIO DESDE LA TERMINAL CON GIT Y GH.....</b>	<b>5</b>
PASO 2.1: Obtener Token de cuenta Github para acceder desde cliente remoto .....	5
PASO 2.2: Instalar Git y GitHub CLI (gh) en Ubuntu .....	7
PASO 2.3: Inicializar el repositorio y añadir los archivos.....	7
PASO 2.4: Registrar tus credenciales de Github.....	7
PASO 2.5: Hacer el primer Commit .....	8
PASO 2.6: Crear el repositorio .....	8
PASO 2.7: Subir los archivos a Github.com con PUSH.....	9
<b>3. CREAR CONTENEDOR JENKINS BÁSICO CON DOCKER-COMPOSE .....</b>	<b>10</b>
PASO 3.1: Crear archivo docker-compose.yml.....	10
PASO 3.2: Inicializar Contenedor de Jenkins.....	10
<b>4. CREAR PIPELINE EN JENKINS PARA CLONAR Y EJECUTAR PRUEBAS .....</b>	<b>11</b>
PASO 4.1: Creacion de nuevo Pipeline .....	11
PASO 4.2: Estructura de un Pipeline Script.....	12

PASO 4.3: Registro de Credenciales de Github en los Jenkins .....	13
PASO 4.4: Script para clonar y ejecutar pruebas unitarias .....	16
PASO 4.5: Consideraciones de un Script que instala paquetes con SHELL .....	17
PASO 4.6 Construir/Ejecutar Manualmente el Pipeline .....	18
<b>5. ACCESO REMOTO A JENKINS MEDIANTE TUNEL HTTP CON NGROK.....</b>	<b>21</b>
PASO 1: Crear Cuenta en ngrok y obtener Authtoken.....	21
PASO 2: Instalar ngrock en ubuntu .....	21
PASO 3: Registrar Credenciales ngrok .....	21
PASO 4: Habilitar tunel HTTP para el puerto 8080 de Jenkins.....	21
<b>6. EJECUTAR PIPELINE MEDIANTE UN WEBHOOKS .....</b>	<b>21</b>

## 1. MÉTODO SUMA CON PYTHON Y PRUEBAS UNITARIAS CON PYTEST

### PASO 1.1: Crear archivo calc.py

#### Comando

```
mkdir pipeline  
cd pipeline  
sudo nano calc.py
```

#### Contenido calc.py

```
def suma(a, b):  
    try:  
        # Convertir a float para ver si el dato es y un numero valido  
        a = float(a)  
        b = float(b)  
        return a + b  
    except ValueError:  
        return "error"
```

### PASO 1.2: Crear archivo de pruebas unitarias test.py

#### Comando

```
sudo nano test.py
```

#### Contenido test.py

```
import pytest  
from calc import suma  
  
def test_suma_numeros():  
    # Prueba 1: suma de dos numeros enteros  
    assert suma(2, 3) == 5  
  
    # Prueba 2: suma de dos numeros decimales  
    assert suma(2.5, 3.5) == 6.0  
  
    # Prueba 3: suma de numero entero y numero decimal  
    assert suma(2, 3.5) == 5.5  
  
    # Prueba 4: manejo de error al recibir un texto  
    assert suma(2, "texto") == "error"
```

### **PASO 1.3: Requerimientos para realizar pruebas unitarias en Ubuntu**

Para hacer las pruebas en la terminal, necesitamos instalar pytest, que es un framework de pruebas unitarias de Python para probar el archivo test.py. **pip** es el gestor de paquetes de Python, y permite instalar, actualizar y gestionar paquetes adicionales de Python como pytest.

**NOTA IMPORTANTE:** Cuando instales paquetes para Python lo puedes hacer de forma global o en un entorno virtual. La instalación global de paquetes en Python los hace accesibles para todos los proyectos en el sistema, pero puede generar conflictos de versiones si diferentes proyectos necesitan versiones distintas de un mismo paquete. En cambio, la instalación en un entorno virtual crea un espacio aislado de dependencias para cada proyecto, evitando conflictos y facilitando la gestión de versiones específicas. Para discontinuar un entorno virtual, basta con eliminar su carpeta, mientras que los paquetes instalados globalmente permanecen en el sistema hasta ser desinstalados manualmente.

#### **Comandos (OPCION INSTALACIÓN GLOBAL)**

```
sudo apt update  
sudo apt install python3-pip  
pip3 install pytest
```

#### **Comandos (OPCION INSTALACIÓN EN UN ENTORNO VIRTUAL)**

```
sudo apt update  
sudo apt install python3 python3-venv  
python3 -m venv venv  
source venv/bin/activate  
pip install pytest
```

[python3 -m venv venv] Crea un entorno virtual llamado "venv" en la carpeta actual

[source venv/bin/activate] Activa el entorno virtual

## PASO 1.4: Ejecucion de pruebas unitarias con pytest

Teniendo los requerimientos instalados , ahora si puedes ejecutar las pruebas unitarias registradas en test.py

### Comando

```
pytest test.py
```

### Asi luce una prueba unitaria exitosa

```
root@servidor:/home/galindo/pipeline# pytest test.py
===== test session starts =====
platform linux -- Python 3.12.3, pytest-7.4.4, pluggy-1.4.0
rootdir: /home/galindo/pipeline
collected 1 item

test.py .

===== 1 passed in 0.00s =====
root@servidor:/home/galindo/pipeline#
```