**📘 Uso de Docker en Pruebas Automatizadas con Python, Selenium, Selenoid, VNC y Percy**

**1. Introducción**

Docker se ha convertido en una herramienta esencial para facilitar el despliegue de entornos de prueba, especialmente combinándolo con frameworks de automatización como **Selenium** y **Python**. Permite tener ambientes de prueba consistentes, escalables y fáciles de administrar.

**2. ¿Docker requiere instalación?**

Sí.  
Es necesario instalar **Docker** en tu máquina. Puedes descargarlo desde Docker Desktop.

**3. Diferencias entre Docker y Docker Compose**

| **Docker** | **Docker Compose** |
| --- | --- |
| Corre contenedores individuales. | Orquesta múltiples contenedores en un solo archivo .yml. |
| Comandos: docker run, docker stop. | Comando: docker-compose up, docker-compose down. |

**4. Uso de Docker Compose para Selenium Grid con VNC (Selenium 3)**

Usando imágenes \*-debug que traen un servidor **VNC** integrado.

yaml

CopiarEditar

version: "3"

services:

selenium-hub:

image: selenium/hub:3.141.59

container\_name: selenium-hub

ports:

- "4444:4444"

chrome-debug:

image: selenium/node-chrome-debug:3.141.59

container\_name: chrome-debug

depends\_on:

- selenium-hub

environment:

- HUB\_HOST=selenium-hub

ports:

- "5900:5900" # VNC Chrome

shm\_size: 2gb

firefox-debug:

image: selenium/node-firefox-debug:3.141.59

container\_name: firefox-debug

depends\_on:

- selenium-hub

environment:

- HUB\_HOST=selenium-hub

ports:

- "5901:5900" # VNC Firefox

shm\_size: 2gb

* **Importante:** Firefox mapea el puerto 5901:5900.

**5. Uso de Docker Compose para Selenium Grid con Selenium 4 (sin imágenes debug)**

En Selenium 4, las imágenes **no incluyen VNC**.

yaml

CopiarEditar

version: "3"

services:

selenium-hub:

image: selenium/hub:4.20.0

container\_name: selenium-hub

ports:

- "4442:4442"

- "4443:4443"

- "4444:4444"

chrome:

image: selenium/node-chrome:4.20.0

container\_name: chrome

depends\_on:

- selenium-hub

environment:

- SE\_EVENT\_BUS\_HOST=selenium-hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

shm\_size: 2gb

firefox:

image: selenium/node-firefox:4.20.0

container\_name: firefox

depends\_on:

- selenium-hub

environment:

- SE\_EVENT\_BUS\_HOST=selenium-hub

- SE\_EVENT\_BUS\_PUBLISH\_PORT=4442

- SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443

shm\_size: 2gb

**6. Alternativa moderna: Selenoid**

**¿Qué es Selenoid?**

* Un servidor de automatización de navegadores liviano basado en Docker.
* Permite lanzar instancias de navegadores en contenedores bajo demanda.
* Permite ver navegadores vía **VNC** y grabar videos.

**Configuración de browsers.json:**

json

CopiarEditar

{

"chrome": {

"default": "118.0",

"versions": {

"118.0": {

"image": "selenoid/vnc\_chrome:118.0",

"port": "4444",

"path": "/",

"tmpfs": {"/tmp": "size=512m"}

}

}

},

"firefox": {

"default": "118.0",

"versions": {

"118.0": {

"image": "selenoid/vnc\_firefox:118.0",

"port": "4444",

"path": "/",

"tmpfs": {"/tmp": "size=512m"}

}

}

}

}

**Docker Compose de Selenoid + Selenoid UI:**

yaml

CopiarEditar

version: "3"

services:

selenoid:

image: aerokube/selenoid:latest-release

container\_name: selenoid

network\_mode: bridge

ports:

- "4444:4444"

volumes:

- "/var/run/docker.sock:/var/run/docker.sock"

- "./browsers.json:/etc/selenoid/browsers.json"

command: [

"-limit", "10",

"-timeout", "3m"

]

selenoid-ui:

image: aerokube/selenoid-ui:latest-release

container\_name: selenoid-ui

network\_mode: bridge

depends\_on:

- selenoid

ports:

- "8080:8080"

environment:

- SELENOID\_URI=http://selenoid:4444

**7. ¿Qué es VNC?**

**Virtual Network Computing (VNC)** permite acceder al navegador en ejecución dentro de un contenedor de forma gráfica.

Clientes recomendados:

* **RealVNC**
* **TightVNC**

**8. Advertencia sobre imágenes \*-debug**

Las imágenes node-firefox-debug y node-chrome-debug están **deprecadas** después de Selenium 4.

**9. ¿Qué es Docker Swarm?**

**Docker Swarm** es la herramienta nativa de Docker para orquestar múltiples contenedores en clústeres.

**Comandos básicos:**

* Iniciar Swarm: docker swarm init
* Desplegar stack: docker stack deploy -c docker-compose.yml nombre
* Escalar servicios usando deploy en Compose.

**Diferencias:**

| **Docker Compose** | **Docker Swarm** |
| --- | --- |
| Solo local. | Clúster distribuido. |
| No usa deploy. | Usa deploy para réplicas. |

**10. ¿Qué es Percy?**

[Percy](https://percy.io/" \t "_new) permite **pruebas visuales automáticas** capturando capturas de pantalla durante las ejecuciones de prueba.

**Planes:**

* Gratuito: Hasta 5,000 capturas.
* Planes pagos disponibles.

**Requiere:**

* Crear cuenta.
* Obtener **TOKEN**.
* Instalar SDK de Percy en el proyecto.

**11. Estructura del Proyecto**

bash

CopiarEditar

test-automation-project/

│

├── docker-compose.yml

├── browsers.json (Selenoid)

├── tests/

│ ├── test\_login.py

│ └── test\_checkout.py

├── requirements.txt

├── README.md

**12. Esquema Visual: Flujo de Herramientas**

plaintext

CopiarEditar

Test Automation Framework (Python + Selenium)

|

v

Selenium WebDriver

|

v

Docker Compose

|

+--> Selenium Hub (o Selenoid)

|

+--> Chrome Node (VNC)

+--> Firefox Node (VNC)

|

+--> [Alternativa] --> Selenoid UI

|

+--> Percy (Capturas Visuales)

**13. 🚀 Flujo práctico de ejemplo: Selenoid + Selenium + Python**

**1. Levantar Selenoid:**

bash

CopiarEditar

docker-compose up -d

**2. Código Python para conectarse a Selenoid:**

python

CopiarEditar

from selenium import webdriver

options = webdriver.ChromeOptions()

options.add\_argument("--no-sandbox")

options.add\_argument("--disable-dev-shm-usage")

driver = webdriver.Remote(

command\_executor="http://localhost:4444/wd/hub",

options=options

)

driver.get("https://example.com")

print(driver.title)

driver.quit()

**3. Ver ejecución en Selenoid UI:**

* Acceder a http://localhost:8080.

**14. 🎯 Ejemplo práctico de integración de Percy**

**1. Instalar Percy Python SDK:**

bash

CopiarEditar

pip install percy-selenium

**2. Código de prueba integrando Percy:**

python

CopiarEditar

from percy import percy\_snapshot

from selenium import webdriver

options = webdriver.ChromeOptions()

options.add\_argument("--no-sandbox")

options.add\_argument("--disable-dev-shm-usage")

driver = webdriver.Remote(

command\_executor="http://localhost:4444/wd/hub",

options=options

)

driver.get("https://example.com")

# Capturar snapshot visual usando Percy

percy\_snapshot(driver, 'Homepage Example')

driver.quit()

**3. Configurar variables de entorno:**

* PERCY\_TOKEN: tu token de proyecto Percy.

**4. Correr las pruebas y ver resultados en la plataforma de Percy.**

**✅ Conclusiones Finales**

* Docker permite montar entornos reproducibles de automatización.
* Puedes usar Selenium Grid clásico o alternativas modernas como **Selenoid**.
* Si quieres pruebas visuales, Percy es una gran adición.
* Con Swarm puedes escalar fácilmente para correr múltiples navegadores simultáneamente.
* TightVNC y RealVNC permiten ver la ejecución de navegadores en vivo si necesitas validar visualmente.