PADUA — Plan de Implementación en Azure VM con Docker

Índice

1. Archivos Generados
2. Cambios en el Script Python
3. Instrucciones Paso a Paso en Azure VM
4. Estrategia de Volúmenes
5. Configurar Cron
6. Puntos Clave de la Arquitectura

Archivos Generados

Archivo	Propósito
`Dockerfile`	Imagen con Python, Chrome, ChromeDriver, Xvfb, PyAutoGUI
`entrypoint.sh`	Inicia Xvfb → valida volúmenes → ejecuta el script Python
`docker-compose.yml`	Orquestación con volúmenes y límites de recursos
`deploy_and_run.sh`	Script de despliegue automático (build + run + logs + limpieza)
`requirements.txt`	Dependencias Python del proyecto
`.dockerignore`	Excluye datos/logs del contexto de build
`test_selenium_browser.py`	**Modificado** para detectar Docker y ajustar Chrome

---

1. Cambios en el Script Python

Funciones de compatibilidad Docker

Se agregaron 3 funciones y un bloque de detección Docker al inicio del script:

```python
def is_docker():
    """Detecta si el script está corriendo dentro de un contenedor Docker."""
    return os.environ.get("RUNNING_IN_DOCKER", "").lower() == "true"


def safe_input(prompt=""):
    """Input seguro que no falla en entornos no-interactivos (Docker, cron)."""
    if is_docker():
        print(f"{prompt} [Omitido en modo Docker]")
        return ""
    try:
        return input(prompt)
```

```
    except (EOFError, OSError):
        print(f"{prompt} [Omitido - entorno no interactivo]")
        return ""
```

Opciones de Chrome específicas para Docker

```
# === OPCIONES ADICIONALES PARA DOCKER / LINUX ===
if is_docker():
    chrome_options.add_argument("--no-sandbox")
    chrome_options.add_argument("--disable-dev-shm-usage")
    chrome_options.add_argument("--disable-gpu")
    chrome_options.add_argument("--window-size=1920,1080")
    chrome_options.add_argument("--disable-extensions")
    chrome_options.add_argument("--disable-infobars")
    chrome_options.add_argument("--remote-debugging-port=9222")
    # NO usamos --headless porque necesitamos Xvfb + PyAutoGUI
```

ChromeDriver en Docker

```
# === MÉTODO DOCKER: ChromeDriver ya instalado en /usr/local/bin ===
if is_docker():
    service = Service("/usr/local/bin/chromedriver")
    driver = webdriver.Chrome(service=service, options=chrome_options)
```

Importante: No se usa `--headless` porque PyAutoGUI necesita una pantalla real (Xvfb la simula).

---

2. Instrucciones Paso a Paso en la Azure VM

Paso 1: Preparar la VM

```
# Conectar a la VM
ssh usuario@tu-azure-vm.eastus.cloudapp.azure.com

# Instalar Docker (si no está instalado)
sudo apt-get update
sudo apt-get install -y docker.io docker-compose-plugin
sudo systemctl enable docker
sudo systemctl start docker
sudo usermod -aG docker $USER

# IMPORTANTE: Cerrar sesión y reconectar para que aplique el grupo docker
exit
ssh usuario@tu-azure-vm.eastus.cloudapp.azure.com
```

Paso 2: Subir el proyecto

```
# Crear estructura en el servidor
sudo mkdir -p /opt/padua
sudo chown $USER:$USER /opt/padua
```

Desde tu máquina Windows (PowerShell):

```
# Subir archivos de infraestructura
scp Dockerfile entrypoint.sh docker-compose.yml deploy_and_run.sh requirements.txt
test_selenium_browser.py .dockerignore usuario@tu-azure-vm:/opt/padua/
```

Paso 3: Crear estructura de datos

```
# En la Azure VM
mkdir -p /opt/padua/data/WEBSITE
mkdir -p /opt/padua/data/DESCARGA
mkdir -p /opt/padua/data/logs
```

Subir el Excel de entrada desde Windows:

```
scp WEBSITE/GUIAS.xlsx usuario@tu-azure-vm:/opt/padua/data/WEBSITE/
```

Paso 4: Construir y ejecutar

```
cd /opt/padua

# Dar permisos de ejecución
chmod +x entrypoint.sh deploy_and_run.sh
```

OPCIÓN A: Con docker compose

```
export HOST_DATA_DIR=/opt/padua/data
docker compose up --build
```

OPCIÓN B: Con el script de despliegue (recomendado para producción)

```
./deploy_and_run.sh
```

OPCIÓN C: Docker run manual directo

```
# Construir la imagen
docker build -t padua-pod:latest .

# Ejecutar el contenedor
docker run --rm \
    --name padua-pod-manual \
    --shm-size=2g \
    -e DISPLAY=:99 \
    -e TZ=America/Bogota \
    -e RUNNING_IN_DOCKER=true \
    -e PYTHONUNBUFFERED=1 \
    -e PYTHONIOENCODING=utf-8 \
    -v /opt/padua/data/WEBSITE:/app/WEBSITE \
    -v /opt/padua/data/DESCARGA:/app/DESCARGA \
    -v /opt/padua/data/logs:/app/logs \
    --memory=4g \
    padua-pod:latest
```

Paso 5: Configurar Cron (ejecución diaria 6:00 AM)

```
# Dar permisos de ejecución (si no se hizo antes)
chmod +x /opt/padua/deploy_and_run.sh

# Editar crontab
crontab -e
```

Agregar esta línea al final del archivo:

```
0 6 * * * /opt/padua/deploy_and_run.sh >> /opt/padua/data/logs/cron.log 2>&1
```

Verificar que se guardó correctamente:

```
crontab -l
```

---

3. Estrategia de Volúmenes (Diagrama)

```
Azure VM (Host)                    Docker Container
═══════════════                    ════════════════
```

```
/opt/padua/
├── data/
│   ├── WEBSITE/                ──► /app/WEBSITE/      (lectura/escritura)
│   │   ├── GUIAS.xlsx              ← Input
│   │   └── GUIAS_RESULTADO.xlsx    ← Output
│   ├── DESCARGA/               ──► /app/DESCARGA/     (escritura)
│   │   └── 20260218/
│   │       ├── 999093684611/
│   │       │   ├── *.pdf
│   │       │   └── *.jpg
│   │       └── .../
│   └── logs/                   ──► /app/logs/          (escritura)
│       ├── padua_20260218_060000.log
│       └── cron.log
├── Dockerfile
├── entrypoint.sh
├── deploy_and_run.sh
├── docker-compose.yml
├── requirements.txt
└── test_selenium_browser.py
```

Los datos persisten en el host — cuando el contenedor muere, los archivos quedan en `/opt/padua/data/`.

---

4. Puntos Clave de la Arquitectura

Aspecto	Solución
Pantalla virtual	`Xvfb :99 -screen 0 1920x1080x24` en `entrypoint.sh`
PyAutoGUI funciona	Sí, porque Xvfb simula un display X11 real
No headless	Correcto, Chrome corre con GUI sobre Xvfb
Shared memory	`--shm-size=2g` evita crashes de Chrome por `/dev/shm` pequeño
--no-sandbox	Necesario porque Docker corre como root por defecto
Logs	Se guardan con `tee` en `/app/logs/` (montado al host)
Limpieza	`deploy_and_run.sh` borra logs >30 días automáticamente
Input interactivo	`safe_input()` evita `EOFError` en Docker/cron
Retrocompatible	El script sigue funcionando en Windows local sin cambios

---

5. Troubleshooting

Chrome crash por shared memory

```
ERROR: session not created: Chrome failed to start
```

Solución: Asegúrate de usar `--shm-size=2g` en el `docker run`.

Xvfb no inicia

```
ERROR: Xvfb no se pudo iniciar
```

Solución: Verifica que no haya otro proceso usando el display `:99`:

```
docker exec -it padua-pod-manual bash
rm -f /tmp/.X99-lock
Xvfb :99 -screen 0 1920x1080x24 &
```

PyAutoGUI no detecta display

```
pyautogui.FailSafeException: PyAutoGUI fail-safe triggered
```

Solución: Verificar la variable `DISPLAY`:

```
docker exec -it padua-pod-manual bash
echo $DISPLAY  # Debe mostrar :99
xdpyinfo | head  # Debe mostrar info del display virtual
```

Ver logs de ejecución

```
# Últimos logs
ls -lt /opt/padua/data/logs/
tail -100 /opt/padua/data/logs/padua_*.log

# Logs del cron
tail -f /opt/padua/data/logs/cron.log
```

Reconstruir imagen desde cero

```
cd /opt/padua
docker build --no-cache -t padua-pod:latest .
```

---

6. Requisitos de la Azure VM

Recurso	Mínimo	Recomendado
vCPUs	2	4
RAM	4 GB	8 GB
Disco	30 GB	50 GB (por los PDFs)
OS	Ubuntu 22.04 LTS	Ubuntu 24.04 LTS
Serie	B2s	B4ms
Red	Outbound HTTPS (443)	Outbound HTTPS (443)

> **Nota:** Asegúrate de que el Network Security Group (NSG) de la VM permita tráfico saliente HTTPS hacia `alertran.latinlogistics.com.co`.

---

7. Resumen de Archivos del Proyecto

```
/opt/padua/
├── Dockerfile              # Imagen Docker completa
├── entrypoint.sh           # Script de inicio (Xvfb + Python)
├── docker-compose.yml      # Orquestación con volúmenes
├── deploy_and_run.sh       # Automatización de build/run/logs
├── requirements.txt        # Dependencias Python
├── .dockerignore           # Exclusiones del build context
├── test_selenium_browser.py # Script principal (modificado)
├── install.md              # Este archivo
└── data/                   # Datos persistentes (host)
    ├── WEBSITE/
    │   └── GUIAS.xlsx      # Input: Archivo con guías a procesar
    ├── DESCARGA/            # Output: PDFs e imágenes descargadas
    └── logs/                # Logs de ejecución
```