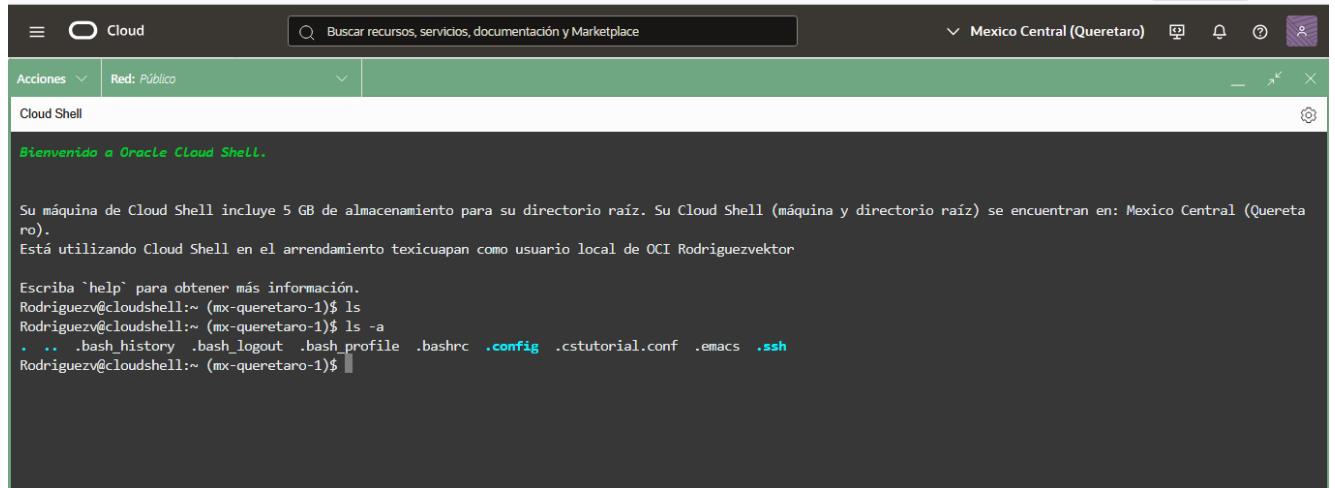


## MANUAL DE DESPLIEGUE DE LA APLICACIÓN FLIGHTONTIME EN OCI

1. Ingresamos a la cloud de shell, buscamos las llaves creadas en la carpeta .ssh



Bienvenido a Oracle Cloud Shell.

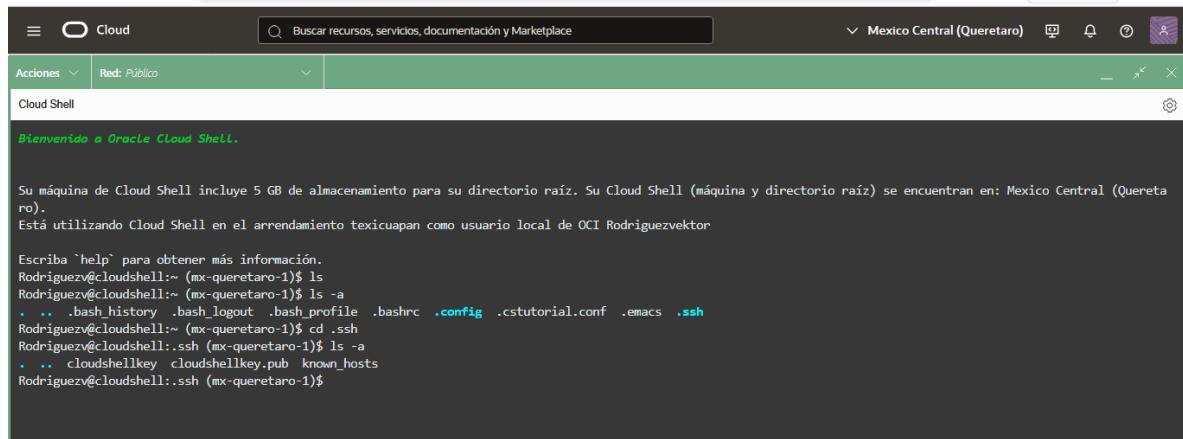
Su máquina de Cloud Shell incluye 5 GB de almacenamiento para su directorio raíz. Su Cloud Shell (máquina y directorio raíz) se encuentran en: Mexico Central (Queretaro).

Está utilizando Cloud Shell en el arrendamiento texicuapan como usuario local de OCI Rodriguezvektor

Escriba `help` para obtener más información.

```
Rodriguezv@cloudshell:~ (mx-queretaro-1)$ ls
Rodriguezv@cloudshell:~ (mx-queretaro-1)$ ls -a
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc  .config  .cstutorial.conf  .emacs  .ssh
Rodriguezv@cloudshell:~ (mx-queretaro-1)$
```

2. Ingresamos a la carpeta .ssh para descargar las llaves , priva y publica



Bienvenido a Oracle Cloud Shell.

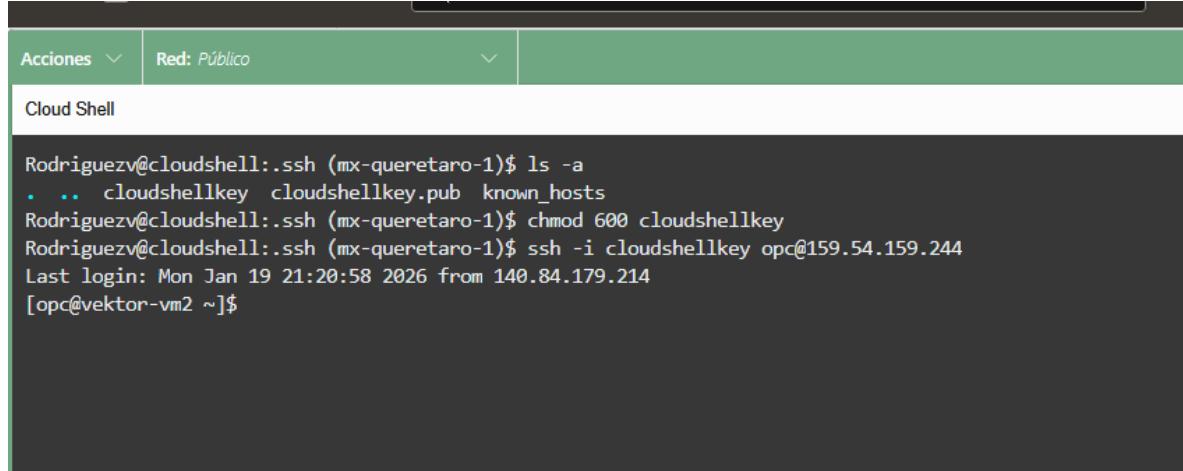
Su máquina de Cloud Shell incluye 5 GB de almacenamiento para su directorio raíz. Su Cloud Shell (máquina y directorio raíz) se encuentran en: Mexico Central (Queretaro).

Está utilizando Cloud Shell en el arrendamiento texicuapan como usuario local de OCI Rodriguezvektor

Escriba `help` para obtener más información.

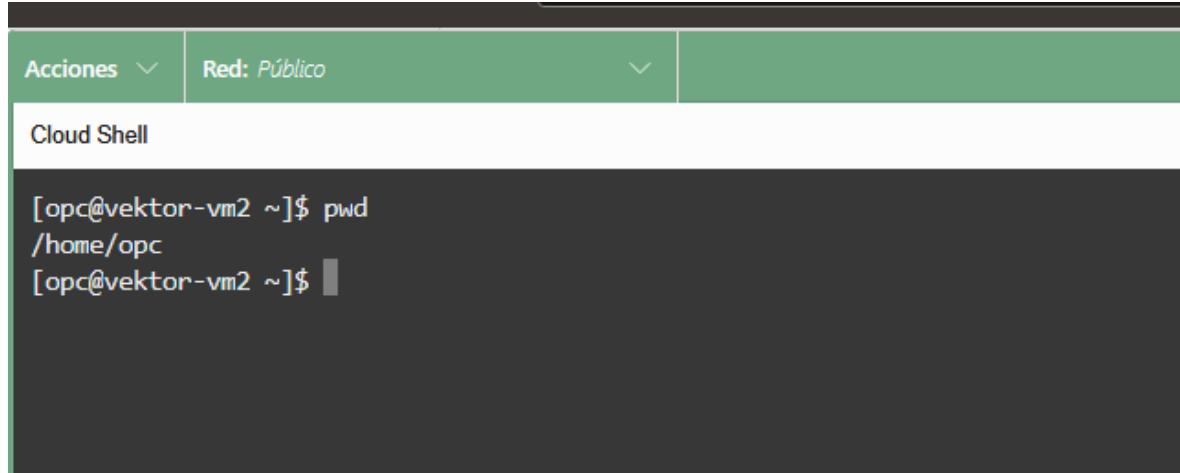
```
Rodriguezv@cloudshell:~ (mx-queretaro-1)$ ls
Rodriguezv@cloudshell:~ (mx-queretaro-1)$ ls -a
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc  .config  .cstutorial.conf  .emacs  .ssh
Rodriguezv@cloudshell:~ (mx-queretaro-1)$ cd .ssh
Rodriguezv@cloudshell:~.ssh (mx-queretaro-1)$ ls -a
.  ..  cloudshellkey  cloudshellkey.pub  known_hosts
Rodriguezv@cloudshell:~.ssh (mx-queretaro-1)$
```

3. Hacemos SSH con la llave privada al servidor



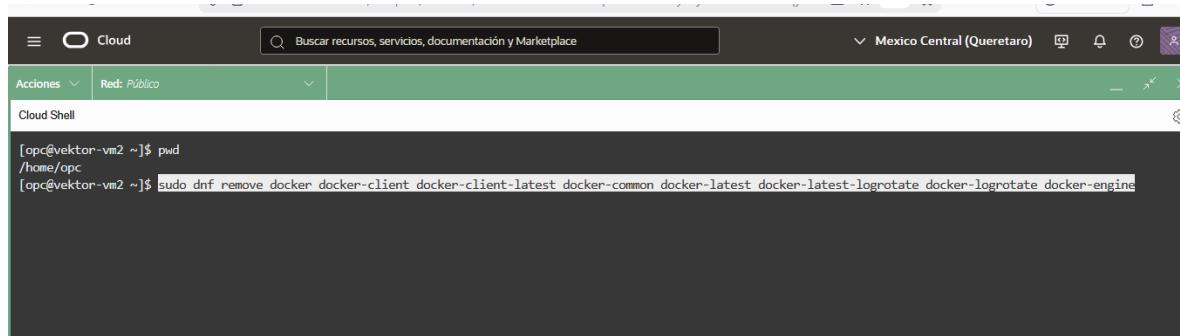
```
Rodriguezv@cloudshell:~.ssh (mx-queretaro-1)$ ls -a
.  ..  cloudshellkey  cloudshellkey.pub  known_hosts
Rodriguezv@cloudshell:~.ssh (mx-queretaro-1)$ chmod 600 cloudshellkey
Rodriguezv@cloudshell:~.ssh (mx-queretaro-1)$ ssh -i cloudshellkey opc@159.54.159.244
Last login: Mon Jan 19 21:20:58 2026 from 140.84.179.214
[opc@vektor-vm2 ~]$
```

4. Hacemos pwd para saber en donde nos encontramos una vez estando en la VM



```
[opc@vektor-vm2 ~]$ pwd
/home/opc
[opc@vektor-vm2 ~]$
```

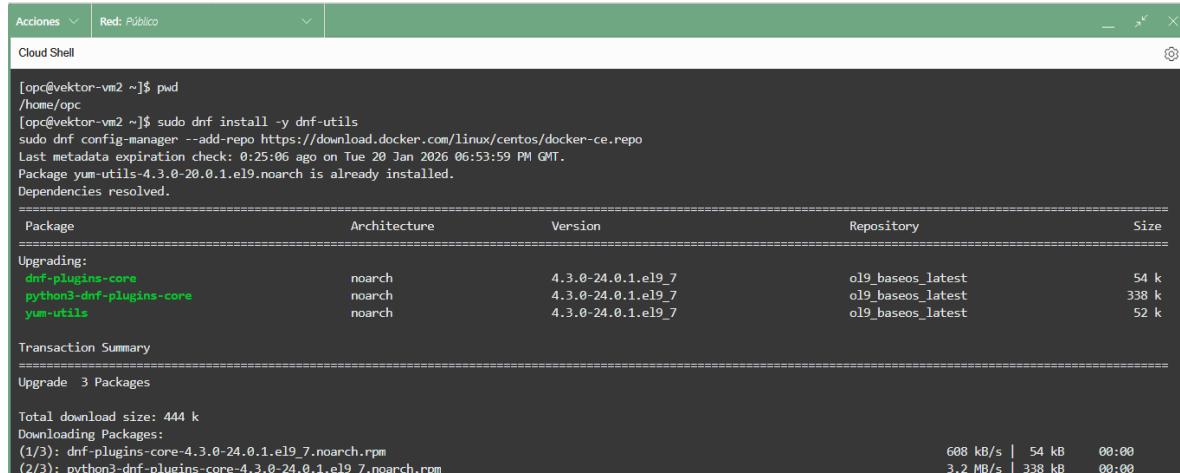
5. Limpiar versiones previas (por seguridad): sudo dnf remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-engine



```
[opc@vektor-vm2 ~]$ pwd
/home/opc
[opc@vektor-vm2 ~]$ sudo dnf remove docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-engine
```

6. Agregar repositorio y herramientas: sudo dnf install -y dnf-utils

sudo dnf config-manager --add-repo <https://download.docker.com/linux/centos/docker-ce.repo>



```
[opc@vektor-vm2 ~]$ sudo dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Last metadata expiration check: 0:25:06 ago on Tue 20 Jan 2026 06:53:59 PM GMT.
Package yum-utils-4.3.0-20.0.1.e19.noarch is already installed.
Dependencies resolved.
=====
 Package          Architecture Version      Repository      Size
=====
Upgrading:
dnf-plugins-core           noarch    4.3.0-24.0.1.e19_7   ol9_baseos_latest  54 k
python3-dnf-plugins-core   noarch    4.3.0-24.0.1.e19_7   ol9_baseos_latest 338 k
yum-utils                 noarch    4.3.0-24.0.1.e19_7   ol9_baseos_latest  52 k
=====
Transaction Summary
=====
Upgrade 3 Packages
Total download size: 444 k
Downloading Packages:
(1/3): dnf-plugins-core-4.3.0-24.0.1.e19_7.noarch.rpm           608 kB/s |  54 kB   00:00
(2/3): python3-dnf-plugins-core-4.3.0-24.0.1.e19_7.noarch.rpm     3.2 MB/s | 338 kB   00:00
```

Elaborado por : Jose Julio Rodriguez

GitHub: JoseBenin82

7. Instalar paquetes: sudo dnf install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

```
[opc@vektor-vm2 ~]$ sudo dnf install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Docker CE Stable - x86_64
Dependencies resolved.
=====
Package           Architecture   Version      Repository    Size
=====
Installing:
containerd.io     x86_64        2.2.1-1.el9      docker-ce-stable 35 M
docker-buildx-plugin x86_64        0.30.1-1.el9      docker-ce-stable 17 M
docker-ce          x86_64        3:29.1.5-1.el9    docker-ce-stable 22 M
docker-ce-cli      x86_64        1:29.1.5-1.el9    docker-ce-stable 8.3 M
docker-compose-plugin x86_64        5.0.1-1.el9      docker-ce-stable 8.1 M
Installing dependencies:
container-selinux   noarch       4:2.237.0-2.el9_6 o19_appstream 73 k
fuse-overlayfs     x86_64        1.15-1.el9       o19_appstream 71 k
pass              x86_64        0*20250512.g8ec1341-4.el9_7 o19_appstream 270 k
pass-selinux       noarch       0*20250512.g8ec1341-4.el9_7 o19_appstream 30 k
Installing weak dependencies:
docker-ce-rootless-extras x86_64        29.1.5-1.el9    docker-ce-stable 3.4 M
=====
Transaction Summary
=====
Install 10 Packages
```

8. Iniciar servicios y dar permisos al usuario opc:

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
sudo usermod -aG docker opc
```

```
newgrp docker
```

```
[opc@vektor-vm2 ~]$ sudo systemctl start docker
sudo systemctl enable docker
sudo usermod -aG docker opc
newgrp docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[opc@vektor-vm2 ~]$
```

## Fase 2: Obtención del Código y Activos

1. Instalar Git: sudo dnf install git -y

```
[opc@vektor-vm2 ~]$ pwd
/home/opc
[opc@vektor-vm2 ~]$ sudo dnf install git -y
Last metadata expiration check: 0:18:00 ago on Tue 20 Jan 2026 07:22:44 PM GMT.
Dependencies resolved.
=====
Package           Architecture   Version      Repository    Size
=====
Installing:
git              x86_64        2.47.3-1.el9_6    o19_appstream 65 k
Installing dependencies:
git-core          x86_64        2.47.3-1.el9_6    o19_appstream 4.7 M
git-core-doc      noarch       2.47.3-1.el9_6    o19_appstream 4.1 M
perl-DynaLoader   x86_64        1:0.17029-7.el9    o19_appstream 24 k
perl-Error        noarch       1:0.17029-7.el9    o19_appstream 57 k
perl-File-Find    noarch       1.37-481.1.el9_6  o19_appstream 24 k
perl-Git          noarch       2.47.3-1.el9_6    o19_appstream 42 k
perl-TermReadKey  x86_64        2.38-11.el9      o19_appstream 42 k
=====
Transaction Summary
=====
Install 8 Packages
```

Elaborado por : Jose Julio Rodriguez

GitHub: JoseBenin82

**2.Clonar el repositorio** Ejecuta esto para descargar el código y guardar la carpeta con el nombre correcto (vektor-ai-backend):

```
git clone https://github.com/VektorAI-Equipo71/vektor-ai.git
```

```
opc@vektor-vm2:~$ git clone https://github.com/VektorAI-Equipo71/vektor-ai.git
Cloning into 'vektor-ai'...
remote: Enumerating objects: 163, done.
remote: Counting objects: 100% (163/163), done.
remote: Compressing objects: 100% (132/132), done.
remote: Total 163 (delta 41), reused 137 (delta 22), pack-reused 0 (from 0)
Receiving objects: 100% (163/163), 4.08 MiB | 8.42 MiB/s, done.
Resolving deltas: 100% (41/41), done.
[opc@vektor-vm2 ~]$
```

**3.Revisamos la carpeta**

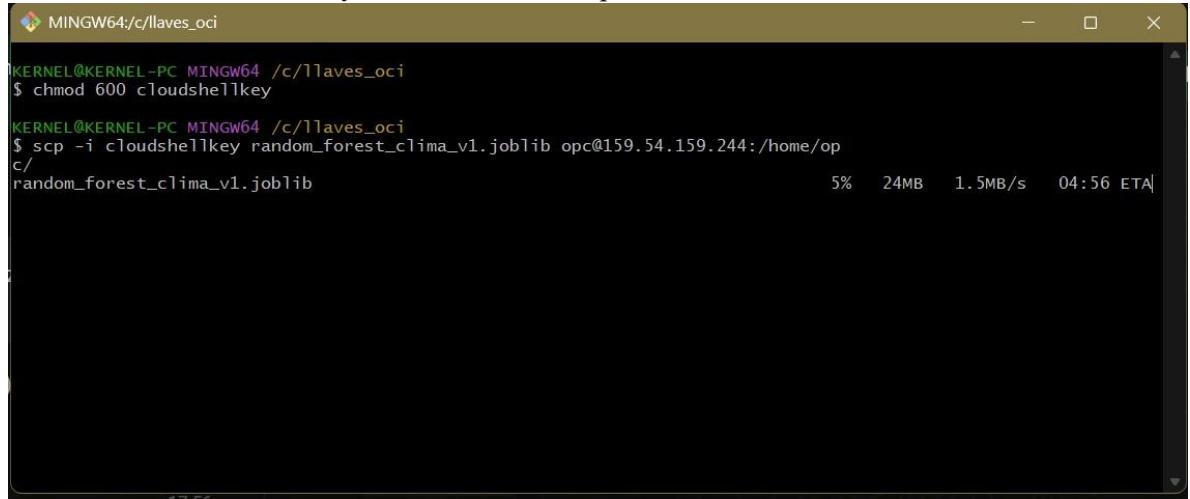
```
opc@vektor-vm2:~/vektor-ai
[opc@vektor-vm2 ~]$ ls
random_forest_clima_v1.joblib  vektor-ai
[opc@vektor-vm2 ~]$ cd vektor-ai
[opc@vektor-vm2 vektor-ai]$ ls
backend docker-compose.yml  docs  frontend  ml-service  notebooks  postman  README.md  scripts
[opc@vektor-vm2 vektor-ai]$
```

**4.Para ello debemos salir de la VM.**

```
MINGW64:/c/Users/julio/Desktop/laves_oci
[opc@vektor-vm2 ~]$ ls
random_forest_clima_v1.joblib  vektor-ai
[opc@vektor-vm2 ~]$ cd vektor-ai
[opc@vektor-vm2 vektor-ai]$ ls
backend docker-compose.yml  docs  frontend  ml-service  notebooks  postman  README.md  scripts
[opc@vektor-vm2 vektor-ai]$ exit
logout
Connection to 159.54.159.244 closed.

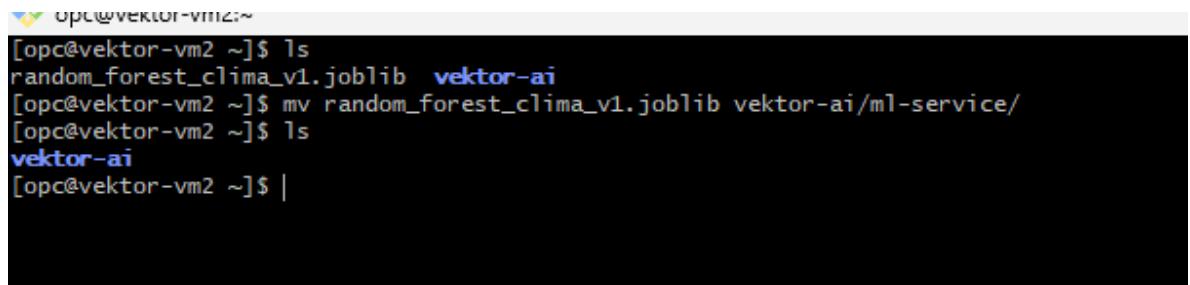
Julius@DESKTOP-FJ8892D MINGW64 ~/Desktop/laves_oci (main)
$ |
```

5. Nos conectamos al servidor y subimos el archivo que tenemos en local del modelo IA



```
KERNEL@KERNEL-PC MINGW64 /c/llaves_oci
$ chmod 600 cloudshellkey
KERNEL@KERNEL-PC MINGW64 /c/llaves_oci
$ scp -i cloudshellkey random_forest_clima_v1.joblib opc@159.54.159.244:/home/opc/
random_forest_clima_v1.joblib                                         5%   24MB   1.5MB/s   04:56 ETA
```

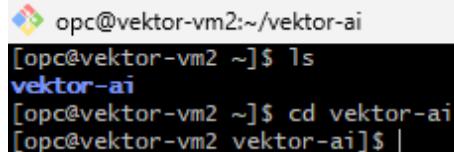
6. Una vez que tenemos el archivo en el servidor procedemos a mover el archivo a la carpeta correspondiente ml-service.



```
[opc@vektor-vm2 ~]$ ls
random_forest_clima_v1.joblib  vektor-ai
[opc@vektor-vm2 ~]$ mv random_forest_clima_v1.joblib vektor-ai/ml-service/
[opc@vektor-vm2 ~]$ ls
vektor-ai
[opc@vektor-vm2 ~]$ |
```

### Fase 3: Despliegue con Docker

1. Entra a la carpeta correcta:



```
opc@vektor-vm2:~/vektor-ai
[opc@vektor-vm2 ~]$ ls
vektor-ai
[opc@vektor-vm2 ~]$ cd vektor-ai
[opc@vektor-vm2 vektor-ai]$ |
```

2. **Levanta el proyecto:** (Este paso va a tardar unos minutos porque descargará y compilará todo. Ten paciencia hasta que te devuelva el control).  
sudo docker compose up -d --build

3. Confirma que todo está encendido:  
sudo docker compose ps

```
[opc@vektor-vm2:~/vektor-ai]$ cd vektor-ai
[opc@vektor-vm2 vektor-ai]$ sudo docker compose ps


| NAME                  | IMAGE                | COMMAND                  | SERVICE    | CREATED        | STATUS                  | PORTS                                       |
|-----------------------|----------------------|--------------------------|------------|----------------|-------------------------|---------------------------------------------|
| flightontime-backend  | vektor-ai-backend    | "java -jar app.jar"      | backend    | 38 minutes ago | Up 38 minutes (healthy) | 0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp |
| flightontime-frontend | vektor-ai-frontend   | "docker-entrypoint..."   | frontend   | 38 minutes ago | Up 38 minutes (healthy) | 0.0.0.0:8081->80/tcp, [::]:8081->80/tcp     |
| flightontime-mi       | Vektor-ai-mi-service | "uvicorn main:app --..." | mi-service | 39 minutes ago | Up 38 minutes (healthy) | 0.0.0.0:8001->8001/tcp, [::]:8001->8001/tcp |
| flightontime-postgres | postgres:15-alpine   | "docker-entrypoint..."   | postgres   | 39 minutes ago | Up 38 minutes (healthy) | 0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp |


```

#### **Fase 4: Conectar Apache (httpd)**

Ahora configuraremos el "puente" para que Apache muestre lo que Docker está ejecutando.  
Elaborado por : Jose Julio Rodriguez  
GitHub: JoseBenin82

Crea el archivo de configuración:

sudo vi /etc/httpd/conf.d/flightontime.conf

1.

```
opc@vektor-vm2:~/vektor-ai$ sudo vi /etc/httpd/conf.d/flightontime.conf
```

2. Pega el contenido:

- Presiona la tecla **i** (para entrar en modo Insertar).
- Copia y pega este bloque exacto:

```
<VirtualHost *:80>
```

```
    ProxyPreserveHost On
```

```
    ProxyRequests Off
```

```
    # Enviar tráfico al Docker (Puerto 8081)
```

```
    ProxyPass / http://localhost:8081/
```

```
    ProxyPassReverse / http://localhost:8081/
```

```
</VirtualHost>
```

```
opc@vektor-vm2:~/vektor-ai
<VirtualHost *:80>
    ProxyPreserveHost On
    ProxyRequests Off

    # Enviar tráfico al Docker (Puerto 8081)
    ProxyPass / http://localhost:8081/
    ProxyPassReverse / http://localhost:8081/
</VirtualHost>

<!-- INSERT -->
```

- Presiona la tecla **Esc**.
  - Escribe **:wq** y presiona **Enter** (para guardar y salir).

```
:wq|
```

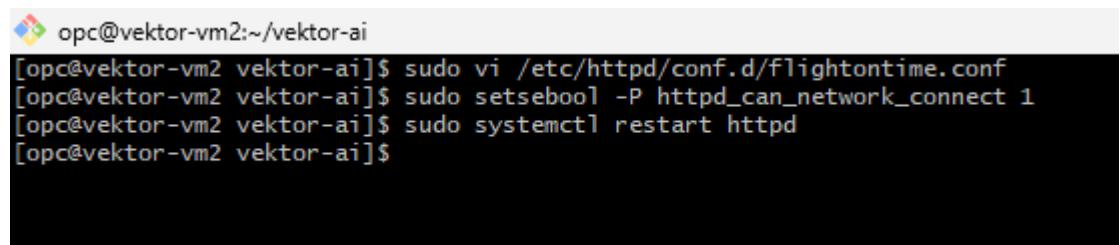
3. Permisos de Red (SELinux) - ¡CRÍTICO!: Sin esto, verás un error 503 aunque todo lo demás esté bien.

```
sudo setsebool -P httpd_can_network_connect 1
```

```
[opc@vektor-vm2 ~]$ sudo vi /etc/httpd/conf.d/flightontime.conf  
[opc@vektor-vm2 ~]$ sudo setsebool -P httpd_can_network_connect 1  
[opc@vektor-vm2 ~]$
```

4. Reinicia Apache:

```
sudo systemctl restart httpd
```

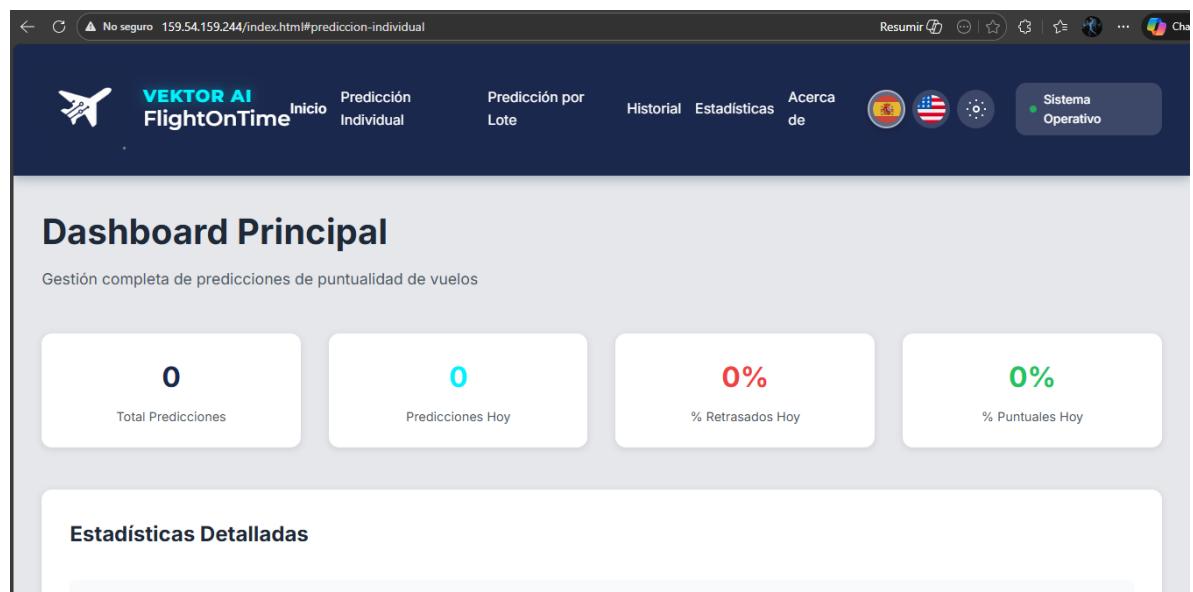


```
opc@vektor-vm2:~/vektor-ai
[opc@vektor-vm2 vektor-ai]$ sudo vi /etc/httpd/conf.d/flightontime.conf
[opc@vektor-vm2 vektor-ai]$ sudo setsebool -P httpd_can_network_connect 1
[opc@vektor-vm2 vektor-ai]$ sudo systemctl restart httpd
[opc@vektor-vm2 vektor-ai]$
```

¡Prueba Final!

Ve a tu navegador y entra a tu IP pública ([http://TU\\_IP\\_PUBLICA](http://TU_IP_PUBLICA)). Debería cargar el sistema.

<http://159.54.159.244>



The screenshot shows a web browser displaying the [FlightOnTime](http://159.54.159.244/index.html#/prediccion-individual) dashboard. The header includes the Vektor AI logo, the FlightOnTime brand name, and navigation links for 'Inicio', 'Predicción Individual', 'Predicción por Lote', 'Historial', 'Estadísticas', and 'Acerca de'. On the right, there are icons for Spanish and English language selection, system status, and a 'Sistema Operativo' button. The main content area features a large title 'Dashboard Principal' and a subtitle 'Gestión completa de predicciones de puntualidad de vuelos'. Below this are four cards with placeholder data: 'Total Predicciones' (0), 'Predicciones Hoy' (0), '% Retrasados Hoy' (0%), and '% Puntuales Hoy' (0%). A section titled 'Estadísticas Detalladas' is partially visible at the bottom.

## ANEXOS

Para que tu navegador pueda ver la página, debemos abrir dos "puertas". Haz estos pasos en orden:

Paso 1: Abrir el Firewall Interno (Linux)

Estás en la terminal, así que hagamos esto primero para permitir tráfico en el puerto 80 (Web).

Ejecuta estos dos comandos:

```
sudo firewall-cmd --permanent --add-service=http
```

```
sudo firewall-cmd --reload
```

```
opc@vektor-vm2:~/vektor-ai
[opc@vektor-vm2 vektor-ai]$ sudo firewall-cmd --permanent --add-service=http
success
[opc@vektor-vm2 vektor-ai]$ sudo firewall-cmd --reload
success
[opc@vektor-vm2 vektor-ai]$
```

## Paso 2: Abrir el Firewall Externo (Oracle Cloud)

**Esta es la causa más probable.** Aunque Linux permita la entrada, la red de Oracle (VCN) la bloquea por defecto. Tienes que salir de la terminal un momento y usar la página web de Oracle.

1. En la consola web de Oracle Cloud (arriba de tu Cloud Shell), busca la barra de búsqueda y escribe: VCN.
2. Selecciona **Virtual Cloud Networks**.
3. Haz clic en el nombre de tu red (suele llamarse vcn-... seguido de la fecha o nombre de tu instancia).
4. En la lista de subredes (Subnets), haz clic en la que dice **Public Subnet**.
5. En la sección "Security Lists" (Listas de seguridad), haz clic en la que aparece ahí (ej. Default Security List...).
6. Verás una lista de reglas. Haz clic en el botón azul "**Add Ingress Rule**" (Agregar regla de entrada).
7. Llena **SOLO** estos campos:
  - **Source CIDR:** 0.0.0.0/0 (Esto permite acceso desde cualquier internet).
  - **IP Protocol:** TCP
  - **Destination Port Range:** 80
8. Haz clic en **Add Ingress Rule** (Agregar regla).

The screenshot shows the 'Agregar reglas de entrada' (Add Ingress Rule) dialog box. It has the following fields filled in:

- Sin estado**: Uncheckable checkbox.
- Tipo de origen**: CIDR, dropdown selected.
- CIDR de origen**: 0.0.0.0/0.
- Protocolo IP**: TCP, dropdown selected.
- Rango de puertos de origen**: (empty).
- Rango de puertos de destino**: 80.
- Descripción**: **Acceso Web HTTP**, highlighted in blue.
- Cancelar** and **Agregar reglas de entrada** buttons at the bottom right.

NOTA: Se tiene que tener instalado previamente httpd en el servidor y activar este.