

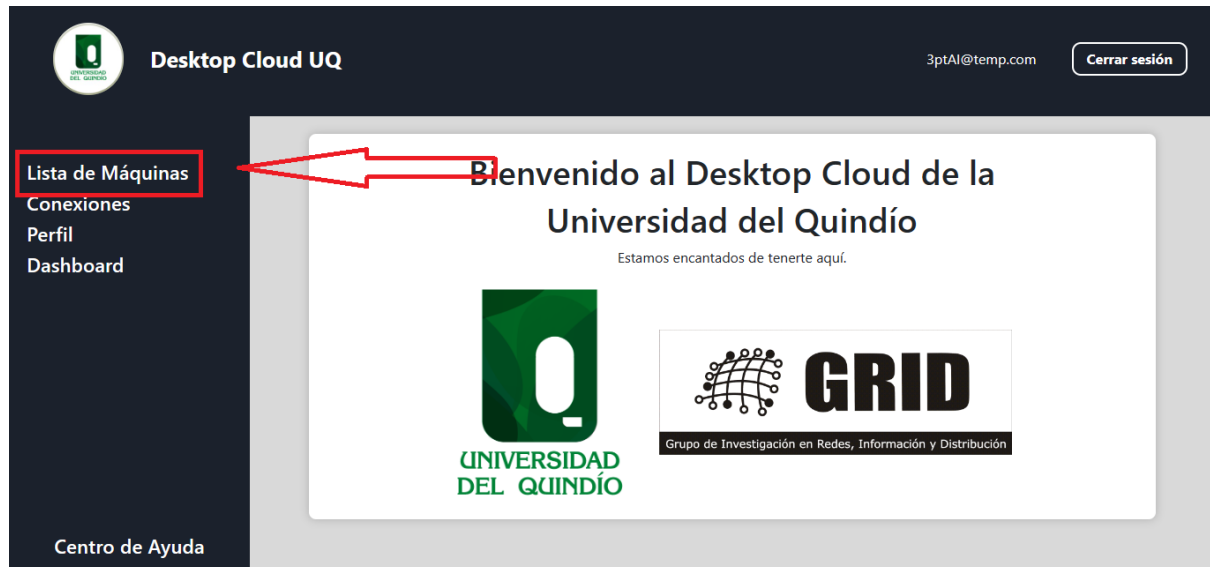
# Laboratorio

Requisitos: Se debe tener instalado Visual Studio Code, en caso de que no lo tenga, la puede descargar de la página oficial desde el siguiente link:

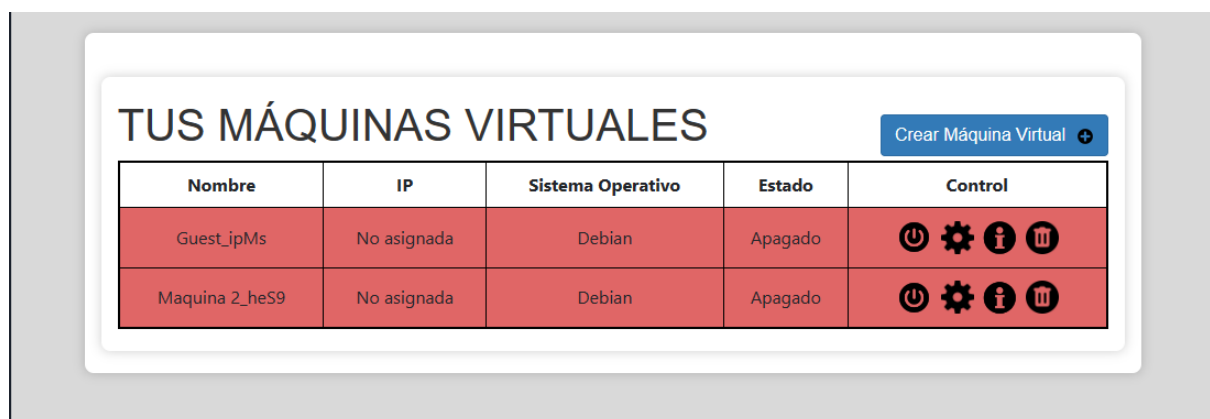
<https://code.visualstudio.com/download>

## Paso 1: Crear una máquina virtual.

En el menú de la izquierda, ingresar a *Lista de Máquinas*:

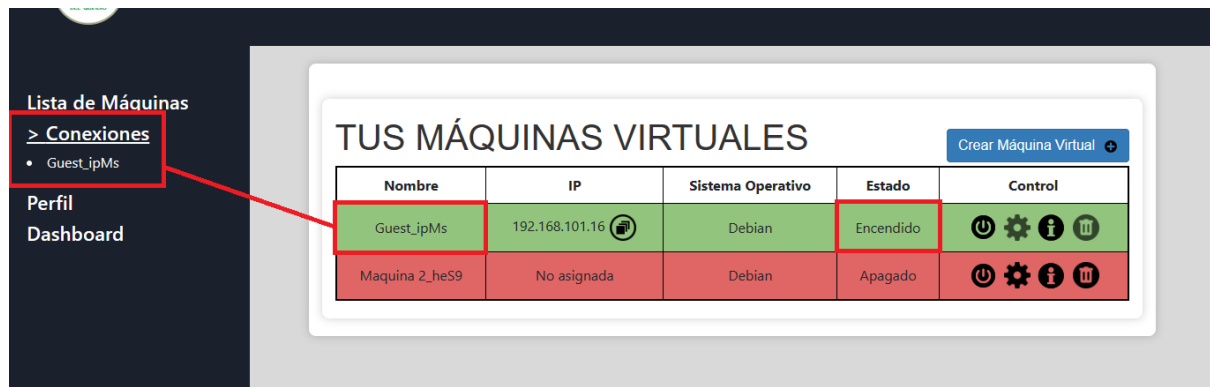


Creamos 2 máquinas virtuales:

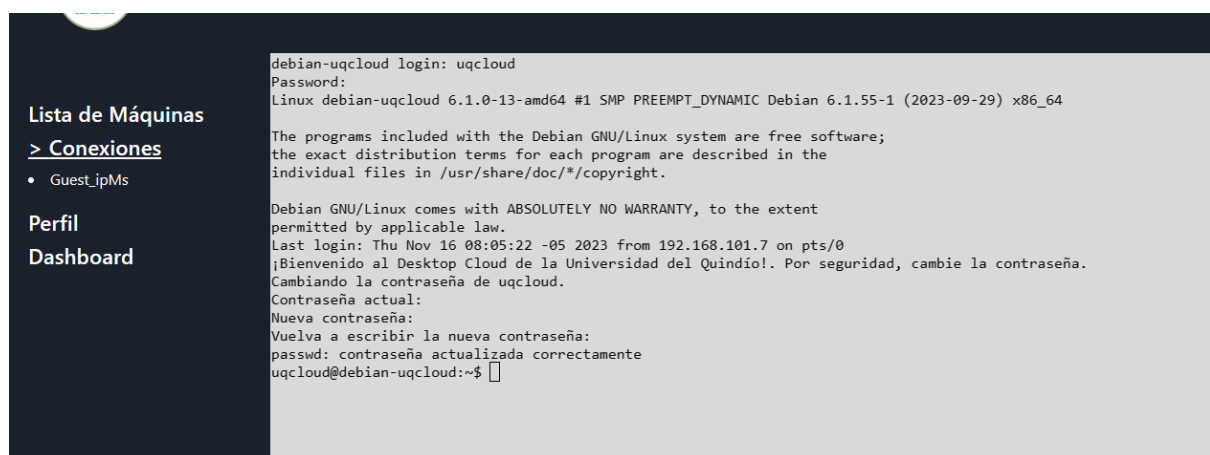


## Paso 2: Instalar Golang

Encendemos una máquina virtual e ingresamos a ella desde el panel izquierdo en la opción *Conexiones*



Ingresamos con el usuario y contraseña por defecto: *uqcloud*



### Paso 3: Instalar Golang

En la máquina que encendimos, escribimos el siguiente comando para instalar golang

- `sudo apt-get install golang`

Verificamos con el comando **go version** que haya quedado instalado:

```

uqcloud@debian-uqcloud:~$ go version
go version go1.19.8 linux/amd64
uqcloud@debian-uqcloud:~$
  
```

### Paso 4: Crear llave SSH

Abrimos CMD desde el computador que estamos usando y escribimos el siguiente comando para crear una llave SSH.

- `ssh-keygen -t rsa -b 2048`

Cuando se esté generando la llave, nos pedirá algunos valores, pero solamente daremos Enter, al final mostrará algo similar a esto:

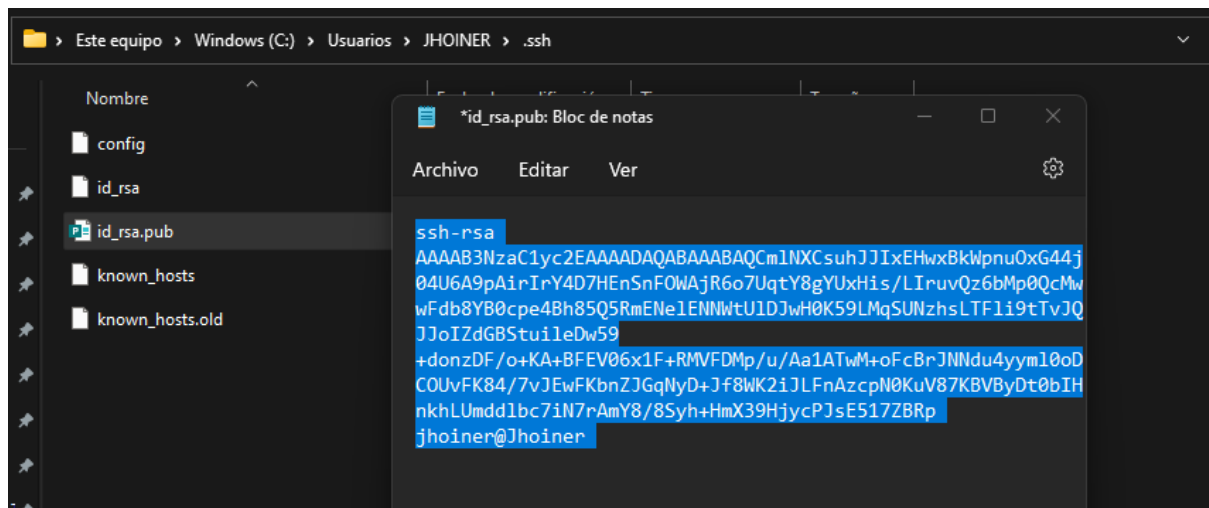
```
C:\Users\JHOINER>ssh-keygen -t rsa -b 2048
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\JHOINER/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\JHOINER/.ssh/id_rsa
Your public key has been saved in C:\Users\JHOINER/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:DquZTZG0aX9awGgM3R811K+cjWdVtnbDzI88/woCi3I jhoiner@Jhoiner
The key's randomart image is:
+----[RSA 2048]-----+
|
|      .+.
|     . . . . . o
|    . . . . .+.o
|   o o o . . .Bo
|  + * S . *.o+
| . + B o = B .
| = E + . .o o
| . X .. . . .
| + +o . .o
+-----[SHA256]-----+
C:\Users\JHOINER>
```

### Paso 5: Copiar la llave pública a la máquina virtual

Vamos a la ruta: C:/Usuarios/**Admin**/.ssh/

Nota: El **nombre de usuario** puede variar. Esto depende del usuario con el cual generamos la llave.

Abrimos con el bloc de notas el archivo **id\_rsa.pub** y copiamos su contenido, así:



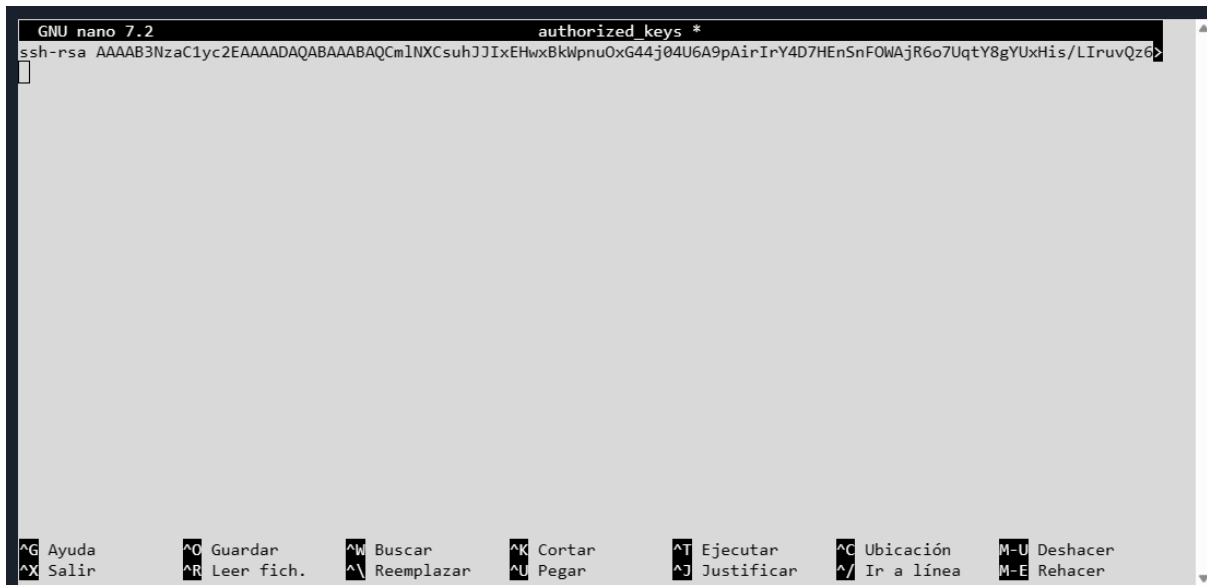
Ahora, vamos a la plataforma Desktop Cloud y en la terminal de la máquina que previamente habíamos encendido, creamos la carpeta `.ssh` e ingresamos a ella, así:

- **`mkdir .ssh`**
- **`cd .ssh`**

```
uqcloud@debian-uqcloud:~$ mkdir .ssh
uqcloud@debian-uqcloud:~$ cd .ssh
uqcloud@debian-uqcloud:~/.ssh$
```

Ahora, con un editor de texto, en este caso nano, vamos a crear el archivo **`authorized_keys`** y presionamos **`ctrl+shift+v`** para pegar el contenido de la llave pública que habíamos copiado desde windows.

Quedará algo similar a esto:

A screenshot of the GNU nano 7.2 text editor. The title bar shows "authorized\_keys \*". The main text area contains a single line of text: "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCMlNXCsuhJJIXEHwxBkwpnu0xG44j04U6A9pAinIrY4D7HEsnFOWAjR6o7UqtY8gYUxHis/LIruvQz6". The bottom status bar displays various keyboard shortcuts: ^G Ayuda, ^O Guardar, ^W Buscar, ^K Cortar, ^T Ejecutar, ^C Ubicación, M-U Deshacer, ^X Salir, ^R Leer fich., ^\_ Reemplazar, ^U Pegar, ^J Justificar, ^/ Ir a línea, M-E Rehacer.

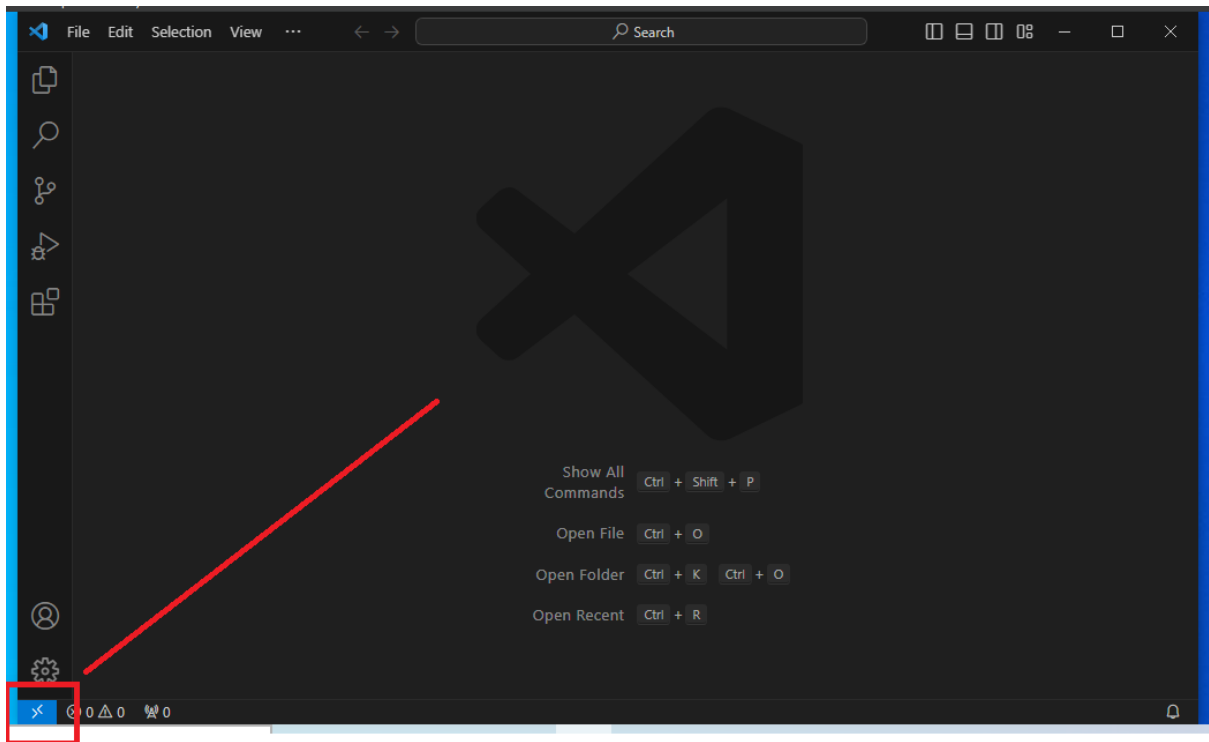
Guardamos y salimos presionando: **ctrl+o** , **Enter**, **ctrl+x**.

Con un **ls** podemos ver el archivo que se creó:

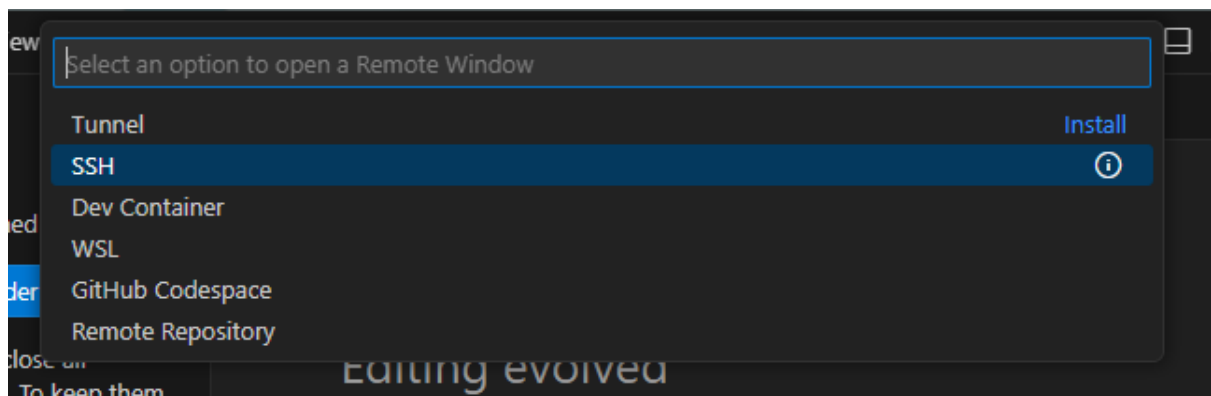
```
uqcloud@debian-uqcloud:~/.ssh$ nano authorized_keys
uqcloud@debian-uqcloud:~/.ssh$ ls
authorized_keys
uqcloud@debian-uqcloud:~/.ssh$
```

## Paso 6: Realizar conexión con Visual Studio Code

Abrimos Visual Studio Code y damos clic en el ícono que aparece abajo a la izquierda de la ventana.

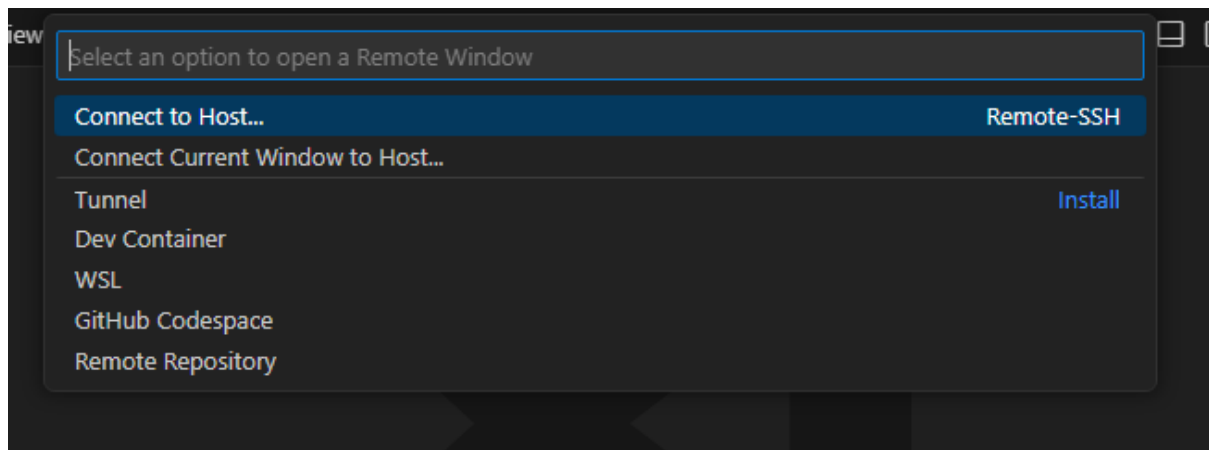


Seleccionamos *SSH*

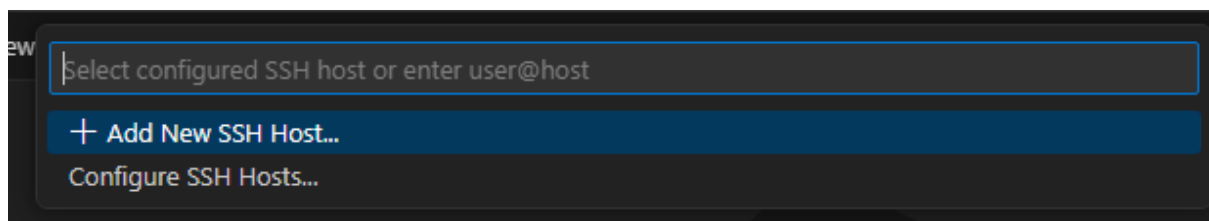


Se instalará automáticamente la extensión para conexión vía SSH

Seleccionamos la opción *Connect to host...*

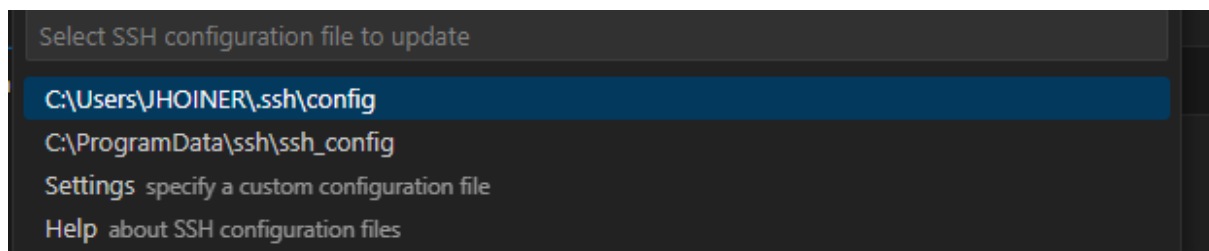


Después en *Add New SSH Host...*

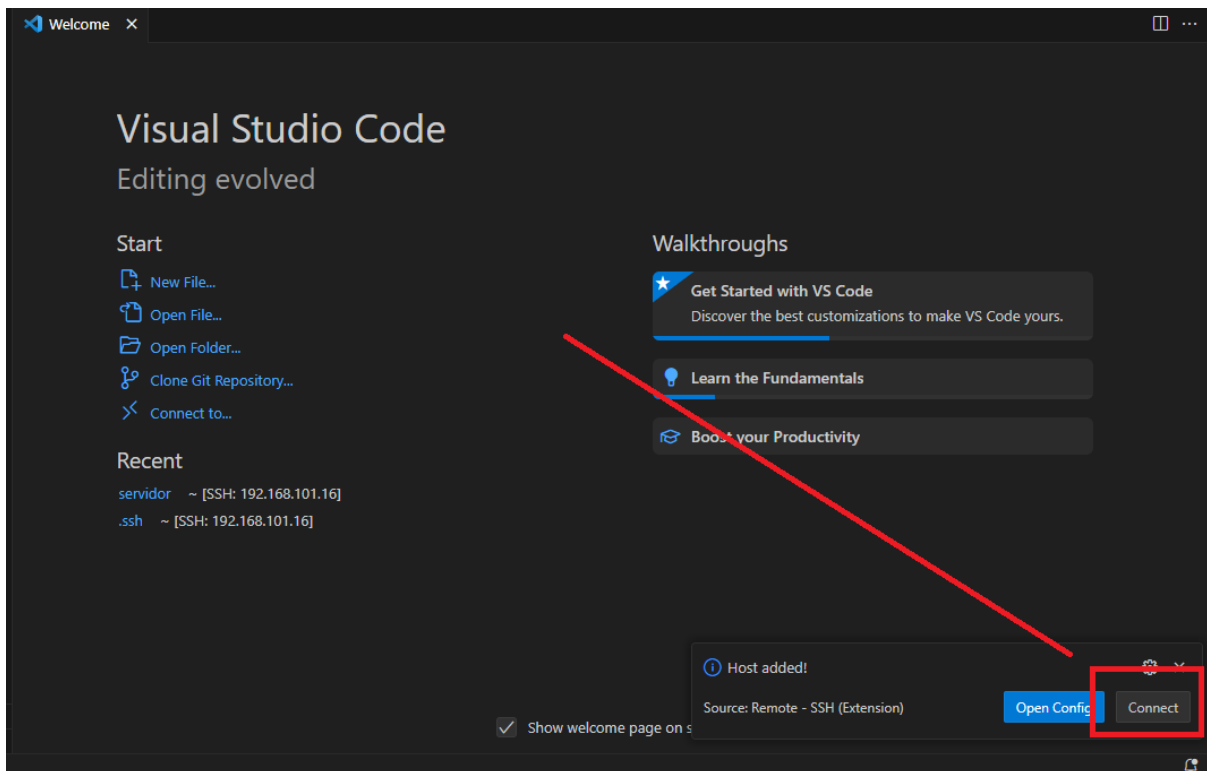


Ingresamos los datos de la conexión, en este caso el usuario uqcloud y la ip de la máquina virtual, el cual se ve algo así: `uqcloud@192.168.101.16`

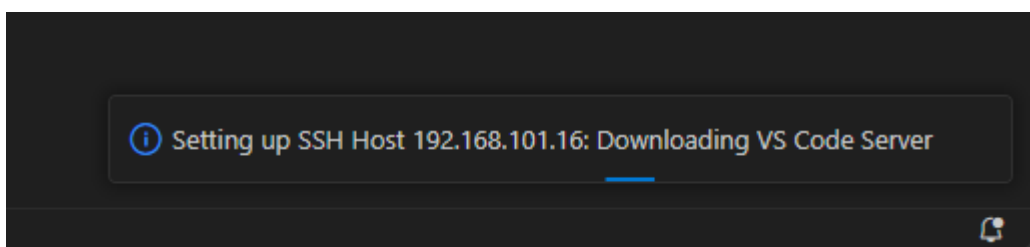
Presionamos Enter y seguido a esto, escogemos el primer archivo de configuración. En este caso el usuario del equipo es *JHOINER*, por eso aparece con esa ruta, en su caso, probablemente el usuario sea *admin*, por lo cual se vería algo como:  
`C:\Users\admin\.ssh\config`



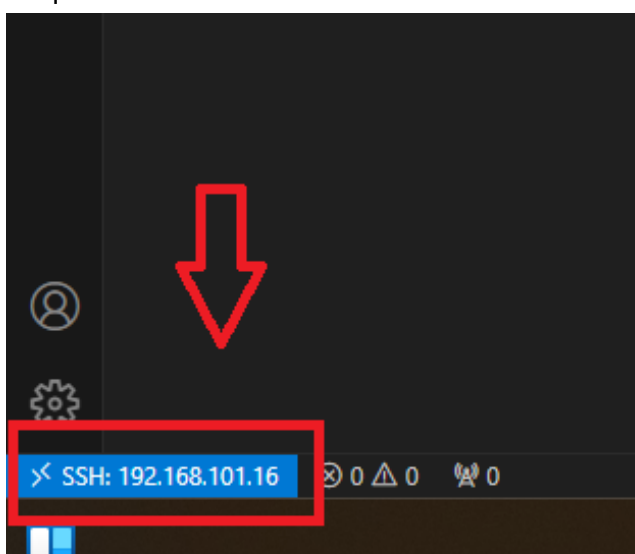
Se abrirá una notificación en la parte de abajo a la derecha. en la cual damos clic en *connect*



Esperamos a que visual studio realice la conexión con la máquina virtual.

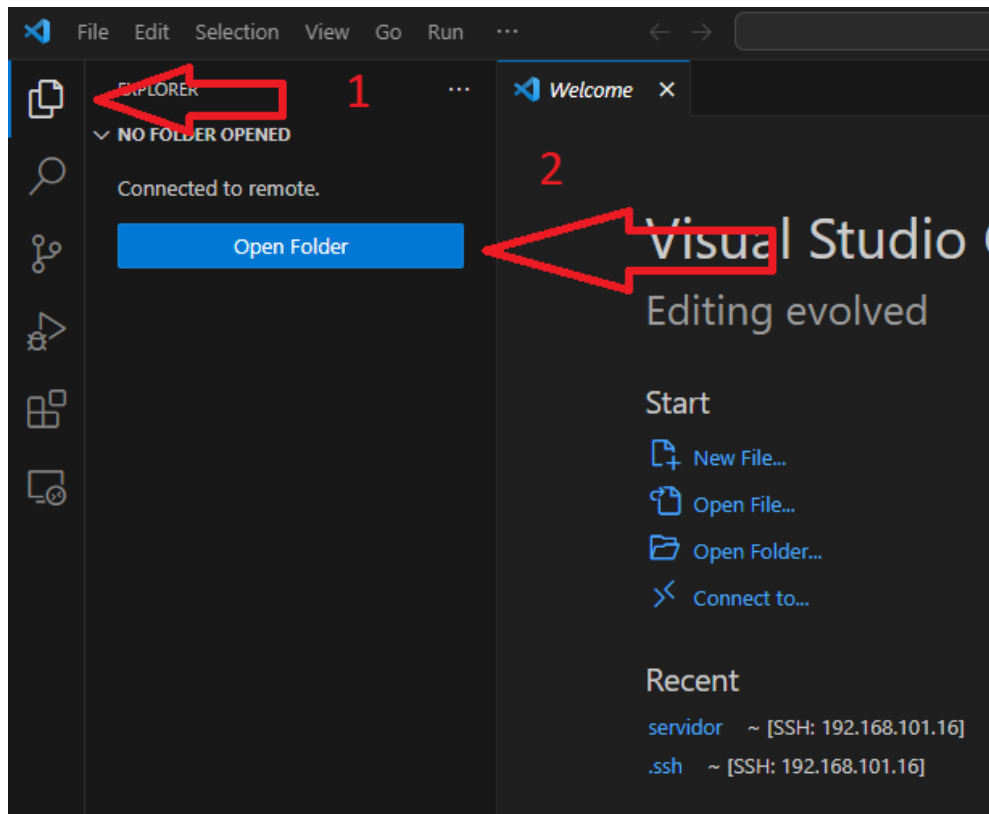


En la parte inferior izquierda, podemos ver que ya estamos conectados vía SSH a la máquina virtual.

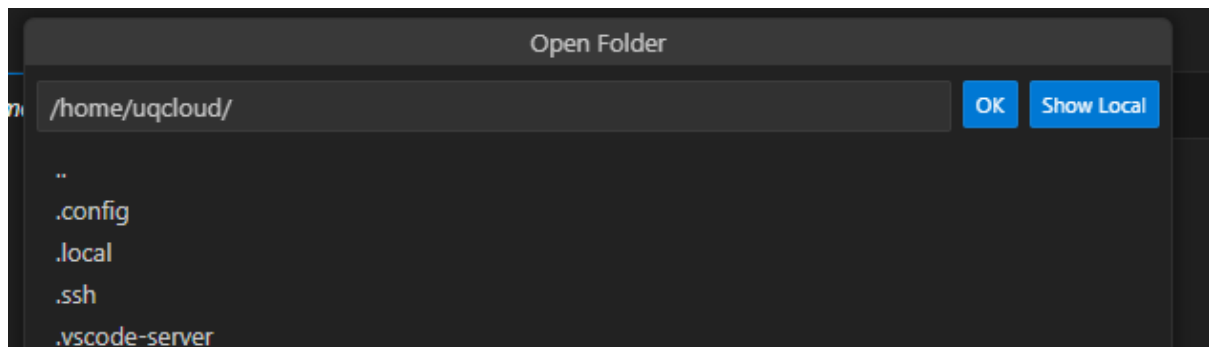




Ahora, vamos al panel izquierdo de visual studio y abrimos el explorador de archivos:



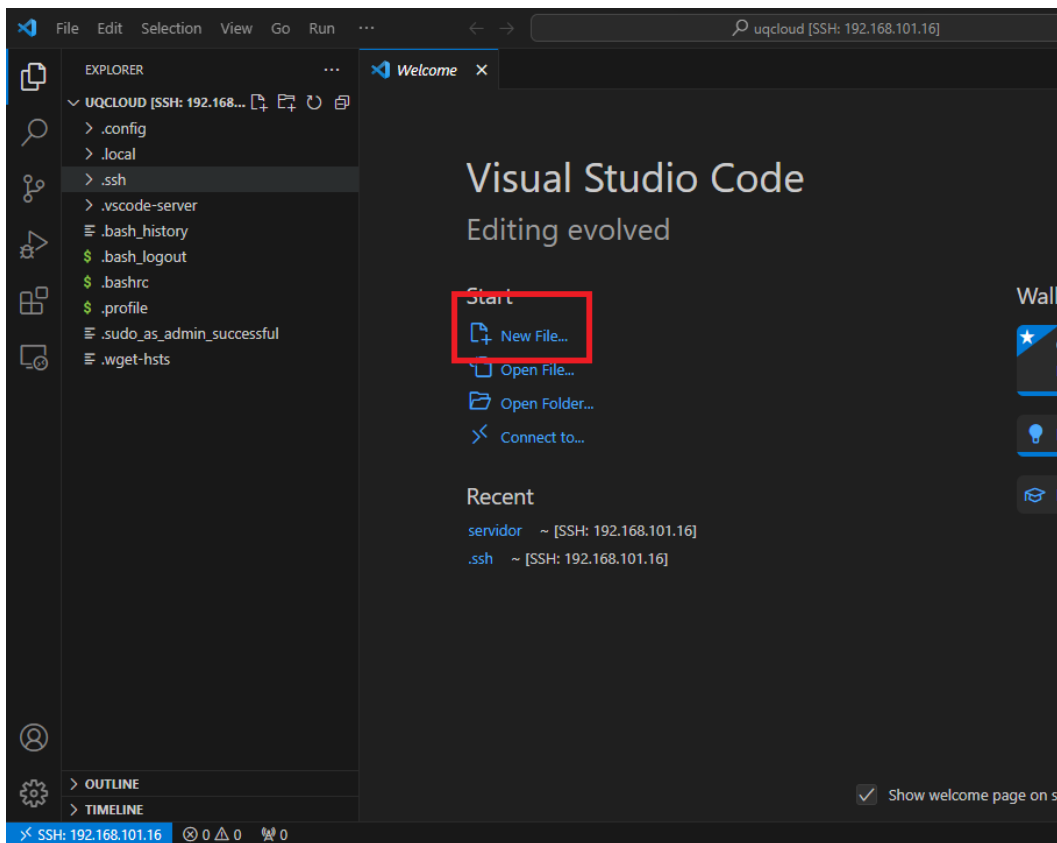
Después, damos clic en *ok*



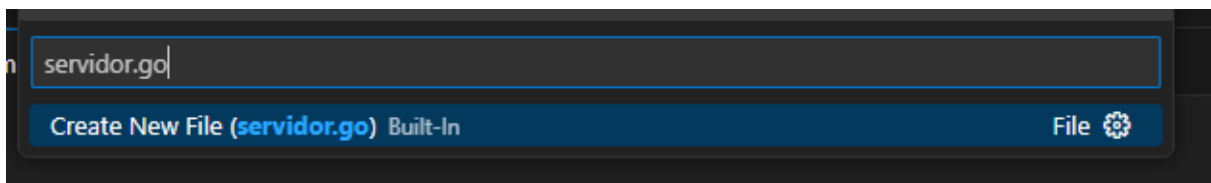
Podemos ver que en el explorador aparecen las carpetas y archivos que hay en la máquina virtual.

### Paso 7: Crear una aplicación WEB con golang

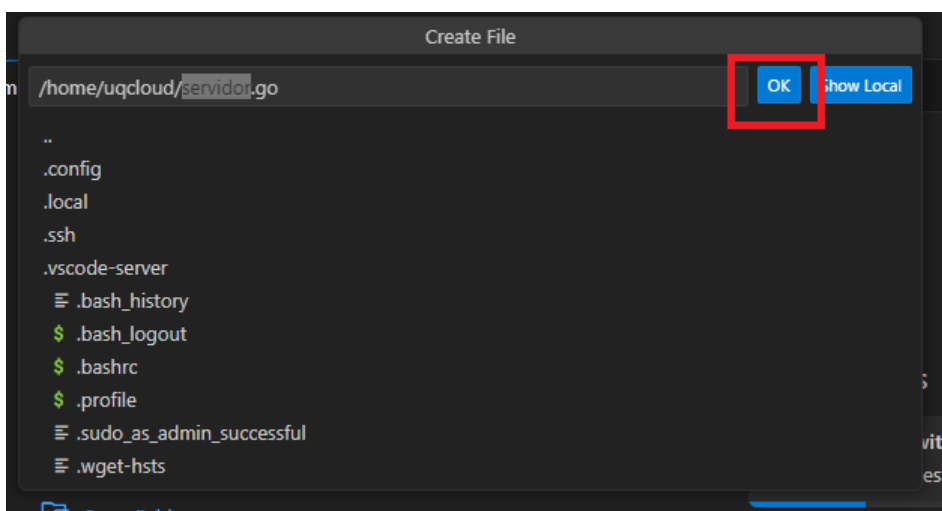
Ahora creamos un nuevo archivo.



El cual llamaremos *servidor.go*



Damos clic en *ok*



Si listamos los archivos en la terminal de la máquina virtual, veremos que se ha creado el archivo *servidor.go* que creamos desde Visual Studio Code.

```
uqcloud@debian-uqcloud:~/ssh$ cd ..
uqcloud@debian-uqcloud:~$ ls
servidor.go
uqcloud@debian-uqcloud:~$
```

En visual studio code, pegamos lo siguiente y guardamos presionando las teclas ctrl+s:

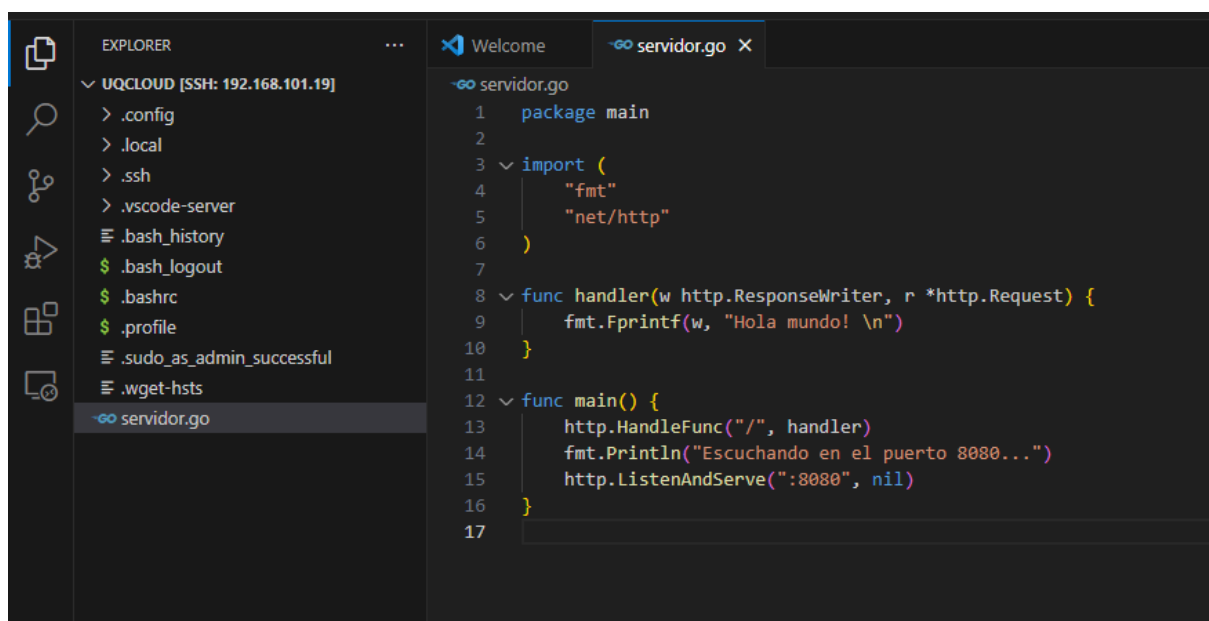
```
package main

import (
    "fmt"
    "net/http"
)

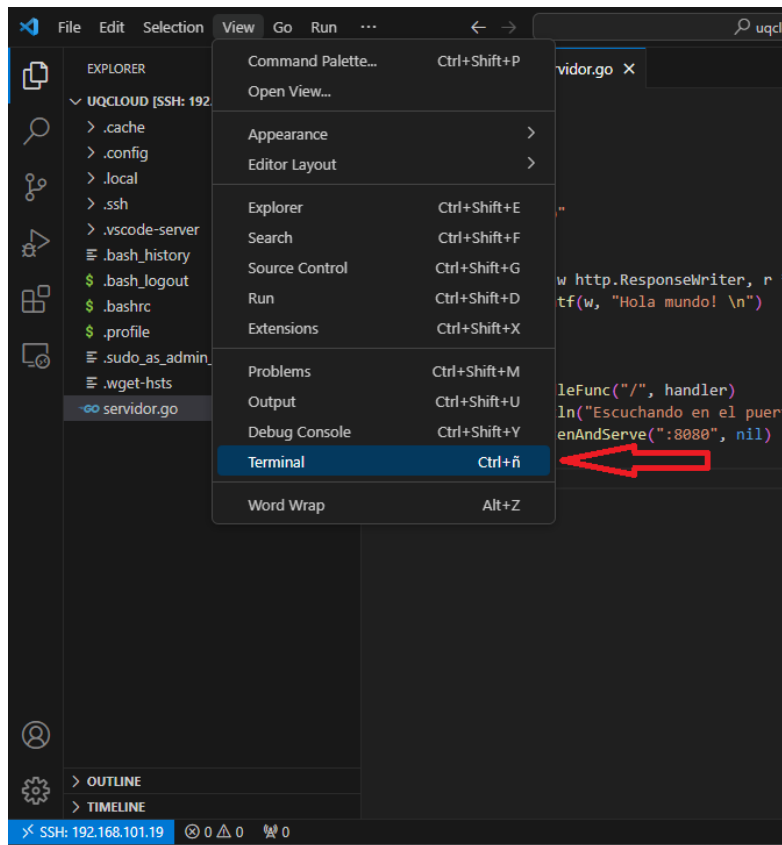
func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hola mundo! \n")
}

func main() {
    http.HandleFunc("/", handler)
    fmt.Println("Escuchando en el puerto 8080...")
    http.ListenAndServe(":8080", nil)
}
```

Quedando así:

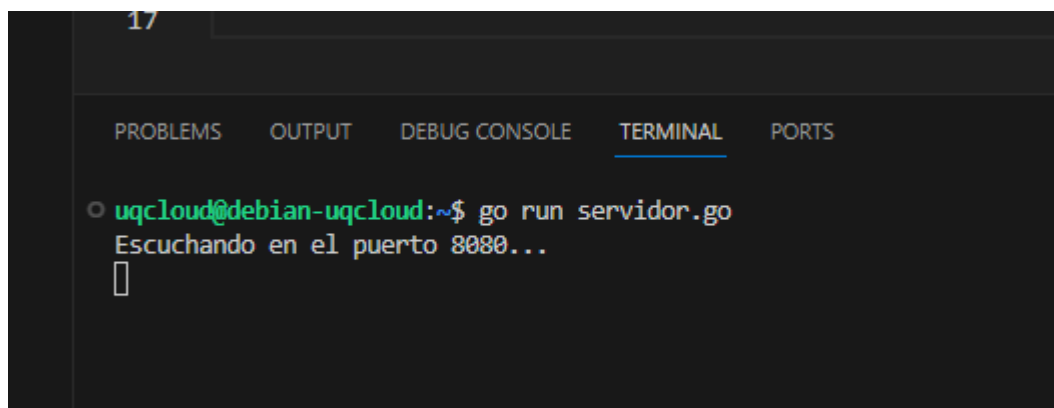


Ahora, abrimos una terminal en Visual Studio Code.



Y escribimos el siguiente comando para poner a correr el servidor.

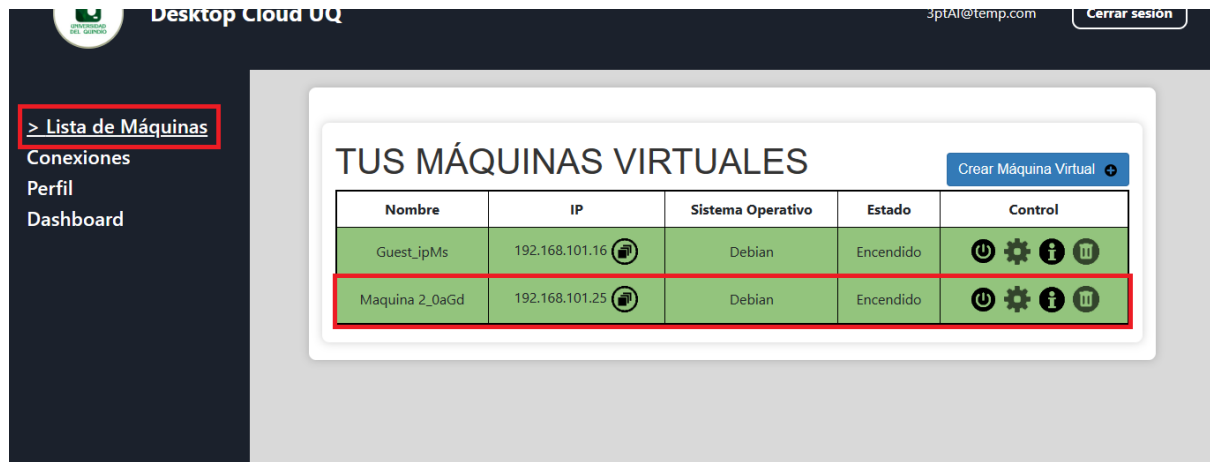
- **go run servidor.go**



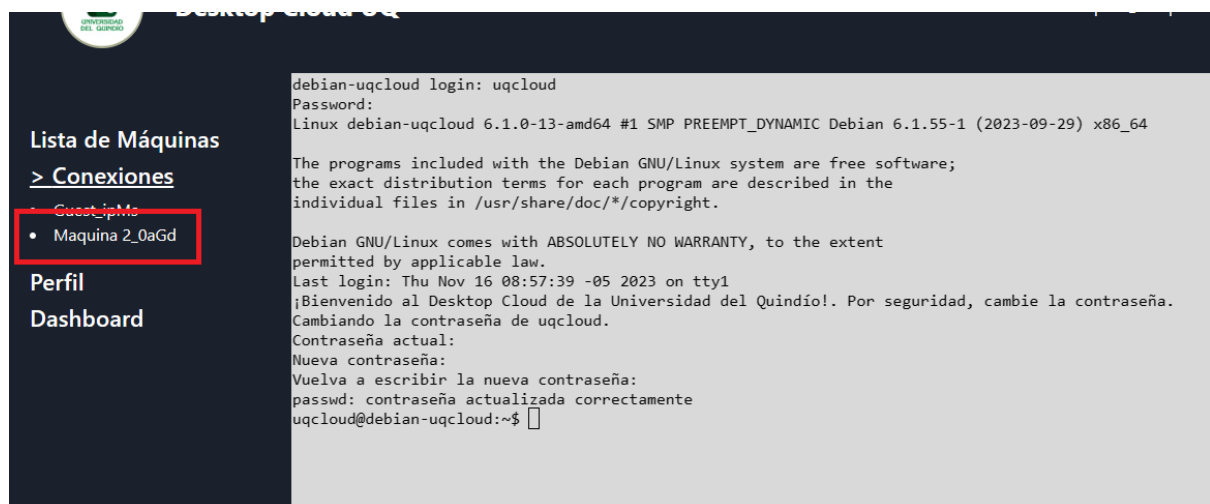
Podemos observar que ya está corriendo el servidor WEB y que está escuchando en el puerto 8080.

## Paso 8: Consumir servicio desde la máquina virtual 2.

En la plataforma Desktop Cloud, en el panel izquierdo, damos clic en Lista de Máquina y encendemos la máquina virtual 2.



Nos conectamos a ella desde el panel de conexiones:



Ya cuando estemos conectados a la máquina 2, procedemos a instalar curl con el siguiente comando:

- **sudo apt-get install curl**

Ahora, escribimos el siguiente comando para consumir el servicio web que creamos en la máquina 1.

- **curl ipMaquina1:8080**

Reemplazamos **ipMaquina1** por la ip de la máquina en donde creamos el servidor web.

```
uqcloud@debian-uqcloud:~$ curl 192.168.101.16:8080
Hola mundo!
uqcloud@debian-uqcloud:~$
```

### Paso 9: Acceder a la máquina 1 vía SSH

Desde la máquina 2, nos conectaremos a la máquina donde está corriendo el servicio web a través de SSH. Para ello, escribimos el comando de conexión SSH

- `ssh uqcloud@ipMaquina1`

Reemplazamos **ipMaquina1** por la ip de la máquina en donde está corriendo el servidor web.

```
uqcloud@debian-uqcloud:~$ ssh uqcloud@192.168.101.16
The authenticity of host '192.168.101.16 (192.168.101.16)' can't be established.
ED25519 key fingerprint is SHA256:cJPw7uVgapxxfnRprHFvdEzHLwloxxTI4mIIE+0go+k.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.101.16' (ED25519) to the list of known hosts.
uqcloud@192.168.101.16's password:
Linux debian-uqcloud 6.1.0-13-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.55-1 (2023-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Nov 16 08:08:01 2023 from 192.168.101.7
uqcloud@debian-uqcloud:~$
```

Ahora, escribimos el comando **ps -a** para ver los procesos que están en ejecución.

```
uqcloud@debian-uqcloud:~$ ps -a
  PID TTY          TIME CMD
 1319 pts/1        00:00:04 go
 1366 pts/1        00:00:00 servidor
 1449 pts/0        00:00:00 ps
uqcloud@debian-uqcloud:~$
```



Podemos ver el proceso que está corriendo con el servidor web.

Vamos a detener la ejecución escribiendo el siguiente comando:

- `kill -9 idProceso`

Reemplazamos **idProceso** por el identificador del proceso, en este caso el id es 1366.

```
1449 pts/0    00:00:00 ps
uqcloud@debian-uqcloud:~$ kill -9 1366
uqcloud@debian-uqcloud:~$ ps -a
  PID TTY          TIME CMD
 1474 pts/0    00:00:00 ps
uqcloud@debian-uqcloud:~$
```

Podemos ver que se detuvo la ejecución.

Además, en la terminal de Visual Studio también se detuvo la ejecución, como se puede ver en la siguiente imagen.

```
uqcloud@debian-uqcloud:~$ go run servidor.go
Escuchando en el puerto 8080...
signal: killed
uqcloud@debian-uqcloud:~$
```

## Paso 10: Modificar el servidor web

Ahora, abrimos con un editor de texto el servidor. En este caso se usará nano.

Cambiamos el texto *Hola Mundo* por *Laboratorio Completado*. Quedando así:

```
GNU nano 7.2                                servidor.go *
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Laboratorio Completado! \n")
}

func main() {
    http.HandleFunc("/", handler)
    fmt.Println("Escuchando en el puerto 8080...")
    http.ListenAndServe(":8080", nil)
}
```

Guardamos y salimos presionando ctrl+o, Enter, ctrl+x

Ahora, ejecutamos el programa:

```
uqcloud@debian-uqcloud:~$ go run servidor.go
Escuchando en el puerto 8080...
```

Abrimos una nueva pestaña en el navegador desde el cual estamos trabajando y en la url ponemos la ip de la máquina donde está corriendo el servicio web (máquina 1) en el puerto 8080.

