

# Registro en Azure y creación de una VM para Mosquitto

## Introducción

Esta guía busca lograr que el estudiante aprenda a usar Azure con su cuenta de estudiante UIS, que conozca lo básico de los servicios que ofrece Azure, que cree una máquina virtual (VM), que le instale todos los recursos necesarios para las próximas prácticas de IoT.

## Objetivos de este taller

Que el estudiante:

- Conquista de Azure, como soluciones de servicios en nube, el cual resulta apropiado para el caso de la UIS, ya que no les exige presentar una tarjeta de crédito.
- Conquista el lenguaje (terminología) apropiado para aprovechar este tipo de servicios
- Implemente un servidor Mosquitto.

## Conocimientos previos

**VM:** Significa Virtual Machine o máquina virtual. En general, una máquina virtual se refiere a una réplica virtualizada de un entorno de hardware de computadora. Esto incluye recursos como CPU, memoria, almacenamiento y red. Una VM se utiliza para ejecutar sistemas operativos y aplicaciones, y puede ser equivalente a una computadora física en muchos aspectos. Las plataformas en la nube, como Google Cloud, AWS, Azure, Alibaba usan ese concepto para referirse a un servicio que ofrecen para que tu cuentes con algo que hace lo mismo que un computador, pero está en la nube de alguna de esas plataformas.

**Instancia:** Este término es conocido en la programación orientada a objetos, donde cada objeto es una instancia de una clase. Pero en las plataformas de nube como Azure, AWS, Google Cloud, tiene una connotación un poco diferente. Aquí hablar de una instancia es hablar de una máquina virtual que se ha creado a imagen de otra que ofrece la plataforma. Imagina que el profesor te da la guía de un taller, tu le sacas una copia y luego trabajas sobre esa copia, pero una copia que conserva dependencia de la original en aspectos como los enlaces, el estilo y demás cosas que son la esencia. Eso significa que tú has instanciado la guía del taller y ahora trabajas en una instancia de la original. A diferencia de las VM tradicionales, las instancias son flexibles en términos de poder reconfigurar sus componentes. En adelante, en este trabajo, cuando usemos el término instancia o VM nos referiremos a lo mismo.

**IP estática:** Una máquina tiene una IP estática si cuando se reinicia mantiene siempre la misma IP.

**IP dinámica:** Una máquina tiene una IP dinámica cuando se reinicia cambia la misma IP.

**CDM:** Es una sigla para la palabra COMMAND. Al mismo tiempo es el nombre que tiene la consola de comandos de Windows.

**Consola:** Es una aplicación en la cual se pueden escribir comandos, es común también llamarle “terminal”. Pero cada empresa puede darle un significado propio. Por ejemplo, cuando se habla de la Consola AWS, se refiere al sitio al que entran las personas cuando se loguean en AWS, donde pueden configurar los servicios de nube que usan, por ejemplo allí pueden crear una VM, encenderla, apagarla, etc. Osea que la consola AWS es realmente el sitio web al que entra un programador para usar los recursos de AWS y lo mismo ocurre con Google Cloud y Azure.

### **Servidor:**

Un servidor es un sistema informático que proporciona recursos, datos, servicios o programas a otros sistemas, conocidos como clientes, a través de una red. Los servidores pueden ser implementados sobre hardware físico, pero también máquinas virtuales o software que realiza tareas específicas para otros programas o dispositivos. En otras palabras, un servidor es un paquete de software instalado en una máquina para crear servicios a los que pueden acceder muchos usuarios desde lugares remotos. Veamos este ejemplo: Tu puedes escribir un archivo HTML que leído por un navegador, despliega una imagen espectacular de tu empresa, pero solo servirá para deleitarte si ese archivo solo está en tu computador, nadie podrá verlo más que tu, necesitas llevarlo a un servidor web que sí tiene la capacidad de ofrecerlo a amplia audiencia. Pero las personas han ido usando este términos en dos ámbitos:

- **Servidor como Software**

Un servidor como software se refiere a una aplicación o programa que presta servicios a otros programas o dispositivos en una red. Estos servicios pueden variar ampliamente dependiendo de la función del servidor. Por ejemplo, si en una VM se instala Mosquitto. Es posible decir que estamos frente a un servidor Mosquitto, también se le puede llamar Broker Mosquitto. Otro ejemplo es que si, adicionalmente, en la VM instaló Apache para publicar páginas web, entonces tendré dos servidores: Mosquitto y Apache.

- **Servidor como Máquina Física o Virtual.** Se refiere a un dispositivo de hardware o una máquina virtual que ejecuta uno o más programas de servidor. Estos servidores pueden alojar múltiples servicios y aplicaciones.

- **Físico:** Es un hardware dedicado que proporciona recursos para ejecutar servicios de servidor. Tiene su propia CPU, memoria, almacenamiento y sistema operativo. Ejemplo: Un servidor rack en un centro de datos, como un Dell PowerEdge o HP ProLiant.
- **Virtual:** Una máquina virtual que emula un servidor físico. Utiliza recursos compartidos de un servidor físico (o varios) mediante tecnologías de virtualización. Ejemplos: máquinas virtuales creadas con VMware, Hyper-V, o

servicios en la nube como Amazon EC2, Microsoft Azure VM dotadas con uno o más servidores de software como Mosquitto, MySQL, Apache.

## Qué es un contenedor

Para entender qué es un contenedor y porqué vale la pena, imagina que en tu familia hay 4 personas pero solo tienen un buen computador para todos, Imaginemos también que una empresa se inventó la manera de lograr que se le puedan conectar a ese computador 4 teclados y 4 pantallas y 4 ratones. Se creará un usuario para cada miembro de la familia. tal y como se hace hoy en el sistema de Windows, donde todos pueden entrar al computador pero cada uno tiene sus cosas propias como aplicaciones, datos, documentos sin interferir con los demás.

Ahora imagina que a un grupo de investigación se le ocurre la idea de aprovechar eso, para que, un usuario se destine para el uso de matlab, otro usuario se destine para llevar toda la parte administrativa del profesor, otro usuario se destine a mantener conectados un sistema de sensores del laboratorio a un software. Con eso el grupo de investigación logra separar cada cosa, por ejemplo, el sistema de sensores del laboratorio no estará en riesgo cuando otro usuario está trabajando en matlab. Pero sobre todo, logra hacer todo eso con un solo computador con lo cual ahorra costos en compra de equipos, mantenimiento, energía.

Los contenedores en la nube tienen similitudes con el ejemplo presentado en los siguientes aspectos:

- El uso de un contenedor es mucho más económico si se compara con una máquina virtual ya que consume menos recursos. Una máquina virtual requiere emular todos los aspectos de un computador, para crear uno o más contenedores se toma lo que ya está emulado, lo que existe y sobre eso crea los contenedores que se requieran.
- Como deducción de lo anterior, los contenedores comparten los mismos recursos de alguna máquina.

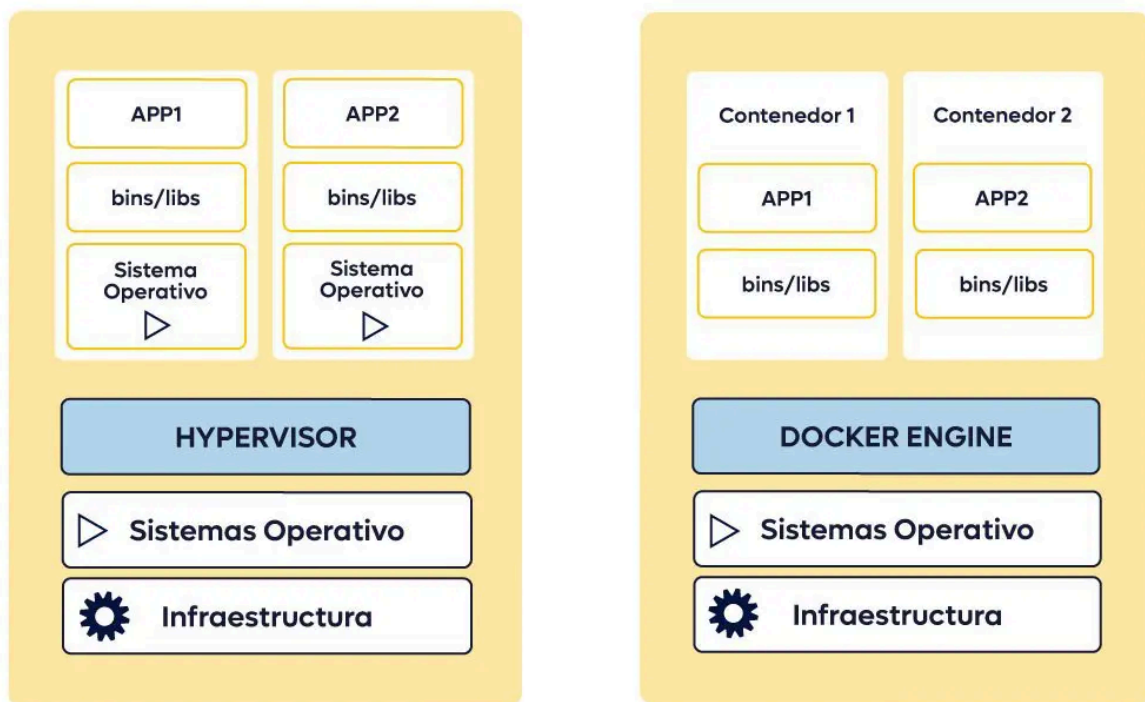
Pero los contenedores se diferencian del ejemplo presentado en los siguientes aspectos:

- Si alguna vez usaste un software en tu computador como VirtualBox, sabrás que es necesario contar con una herramienta como esa para poder correr máquinas virtuales. Pues bien, Windows también tiene herramientas propias para manejar múltiples usuarios. Pero más allá, yendo a lo que hacen los operadores de nube, para poder correr contenedores también se requiere un software especial, por ejemplo Docker. Pero el Docker no es exclusivo de los proveedores de nube, en principio, cualquier persona puede adquirir un servidor físico e instalarle Docker para lograr un efecto similar a lo que se logra con los operadores de nube - poder tener muchos contenedores en una misma máquina. Imagina que tienes en tu computador instalado el Docker. Si necesitas algo dedicado solo para manejar matlab lo puedes instalar en el computador o en un contenedor. Si lo haces en un contenedor, usualmente en ese contenedor te ocuparás de lo mínimo necesario para que corra matlab, por ejemplo, no tiene sentido tener allí power point, word, un navegador y muchísimas cosas más. No, solo te ocuparás de que tenga matlab. Pero, supongamos que no deseas saber nada de Docker, sino usar una aplicación como VirtualBox para crear una VM, lograrás también tener allí matlab, sin problema, pero a costa de mayores costos computacionales y de disco porque una VM emula todos los elementos de un computador y sistema operativo.

- En el caso de la nube, muchas veces ni nos enteramos si por ejemplo Azure usa o no VirtualBox para crear las VM o si usa Docker para crear contenedores, tampoco nos enteramos cuál es la máquina que soporta los contenedores. Ellos simplemente están allí para que los usemos. Tú decides que usar en función de lo que implique pagar por el uso de esos recursos, la comodidad para encapsular servidores separados y de pronto prefieras que todos los servidores estén en una misma VM. También es posible que si eres desarrollador pienses de una manera que facilita tu trabajo usando contenedores con cosas encapsuladas, si eres una empresa quizá prefieras tener todo en una misma VM.
- Existe una amplia base de datos de imágenes de contenedores ya listos para las diversas necesidades. Por ejemplo, para el caso que nos atañe existe una imagen de contenedor ya listo para Node-RED. <https://www.docker.com/> parece ser el servicio estrella para acceder a una amplia gama de imágenes de diversos contenedores. Siguiendo con el ejemplo, contar con un servidor listo para Node-RED puede ser tan sencillo como oprimir unos cuantos botones.
- Imagina que tu aprendiste en una empresa a usar MySQL combinado con otros recursos para crear páginas Web. Eres muy hábil en la creación de soluciones, pero un día te separas de la empresa para independizarte y luego, con horror descubres que no sabes instalar todo lo necesario para realizar el mismo trabajo por tu cuenta. Qué lindo sería para ti poder oprimir un botón y contar con un contenedor con todo instalado y super probado por grandes expertos en instalación, cierto? y qué importa si ese contenedor solo tiene MySQL y si es otro el contenedor que tiene lo de las páginas Web, eso no importa, porque en la nube todo se interconecta por URLs o direcciones IP.

## Comparación entre un máquinas virtuales y contenedores

La comparación se ilustra en la siguiente figura. A la izquierda las máquinas virtuales (VM), a la derecha los contenedores. **Vemos básicamente que cada VM tiene su propio sistema operativo y puede ser diferente al del computador físico. Los contenedores se apoyan todos en el sistema operativo del computador físico.**



Para el caso de la figura de la izquierda, supongamos que tenemos una máquina física sobre Linux, le montamos un hipervisor para poder crear allí máquinas virtuales. Una de esas máquinas virtuales es de Ubuntu, la otra es de Windows. Para el caso de la figura de la derecha, supongamos que tenemos la misma máquina física con Linux, en lugar de un hipervisor le hemos instalado Docker. Podremos crear allí un ambiente para GNU Radio, otro para MySQL, pero siempre sobre Linux.

## Ejemplos de problemas que se solucionan con contenedores

A continuación se busca despertar la imaginación del lector sobre las inmensas posibilidades que abren los contenedores. Para ello se presentan ejemplos de problemas que se solucionan fácilmente mediante el uso de contenedores

### Caso 1. Despliegue de aplicaciones con diferentes exigencias de seguridad

Imagina que tienes una VM dedicada a Mosquitto (es decir, un servidor Mosquitto), y has configurado reglas de acceso muy flexibles para que cualquier dispositivo, desde cualquier IP, como una ESP32, pueda enviar datos al servidor. ¿Qué pasaría si en esa misma VM instalamos un servidor MySQL para gestionar bases de datos? La respuesta es que, al solucionar un problema para Mosquitto, se podría crear un problema de seguridad para MySQL, ya que las reglas de acceso flexibles necesarias para Mosquitto podrían comprometer la seguridad de MySQL.

Una solución sería implementar MySQL en otra máquina con reglas de seguridad específicas. Sin embargo, para reducir costos, podríamos crear un contenedor para MySQL. Esto permite que MySQL tenga su propio entorno aislado y seguro, a pesar de que incluso podría estar en la misma máquina física, aunque en el caso de Azure nunca se sabe en qué máquina están los contenedores, ni te importa. Azure proporciona una capa de abstracción que oculta estos detalles, permitiendo a los usuarios centrarse en el despliegue y gestión de sus aplicaciones sin preocuparse por la infraestructura física específica. Con el uso de contenedores se logra el aislamiento (otros le llaman encapsulación pero este término no es tan correcto), el cual se traduce en seguridad.

Es importante analizar bien la situación, porque también puede tener más sentido montar Mosquitto como un contenedor y que los demás servidores con necesidades comunes de seguridad estén en una VM o en contenedores separados.

## **Caso 2: Despliegue de Aplicaciones con Dependencias Conflictivas**

Imagina que tienes una aplicación web que utiliza Node.js y otra aplicación que utiliza Python con Django. Ambas aplicaciones deben ejecutarse en el mismo entorno de desarrollo, pero tienen dependencias conflictivas.

Instalar todas las dependencias necesarias para ambas aplicaciones en la misma VM puede llevar a conflictos de versiones y problemas de compatibilidad, complicando el mantenimiento y el desarrollo.

Usar contenedores para cada aplicación. Cada contenedor puede tener su propio entorno aislado con las versiones específicas de Node.js y Python necesarias. Esto elimina los conflictos de dependencias y facilita la gestión y actualización de las aplicaciones.

Los contenedores permiten el aislamiento de las dependencias, evitando conflictos y simplificando la gestión del entorno de desarrollo.

## **Caso 3: Escalabilidad de Servicios Independientes**

Tienes una aplicación de comercio electrónico con varios componentes, como el front-end, el back-end, el servicio de pagos y el servicio de inventario.

Cada componente tiene diferentes requisitos de escalabilidad. Por ejemplo, durante las temporadas de alta demanda, el servicio de pagos y el servicio de inventario pueden necesitar escalar rápidamente, mientras que el front-end puede no necesitar tantos recursos adicionales.

Desplegar cada componente de la aplicación en contenedores separados. Esto permite escalar cada servicio de manera independiente según sus necesidades específicas, sin tener que escalar toda la aplicación como un único bloque.

La escalabilidad independiente mejora la eficiencia de recursos y reduce costos, al tiempo que garantiza que los servicios críticos puedan manejar el aumento de la carga.

## Caso 4: Entornos de Prueba y Desarrollo

Tu equipo de desarrollo necesita replicar el entorno de producción para probar nuevas características y correcciones de errores.

Recrear el entorno de producción exacto en una VM tradicional puede ser costoso y llevar mucho tiempo, especialmente si se requiere configuraciones específicas y dependencias complejas.

Usar contenedores para crear réplicas exactas del entorno de producción. Con Docker, por ejemplo, puedes definir todas las configuraciones y dependencias en un archivo Dockerfile y crear contenedores idénticos al entorno de producción en cuestión de minutos.

La facilidad y rapidez para crear entornos de prueba y desarrollo idénticos al de producción, lo que mejora la calidad del software y acelera el ciclo de desarrollo.

## Caso 5: Migración y Portabilidad

Necesitas mover una aplicación desde un entorno on-premises a la nube, o de un proveedor de nube a otro.

Las diferencias en las configuraciones del entorno y las dependencias pueden hacer que la migración sea complicada y propensa a errores.

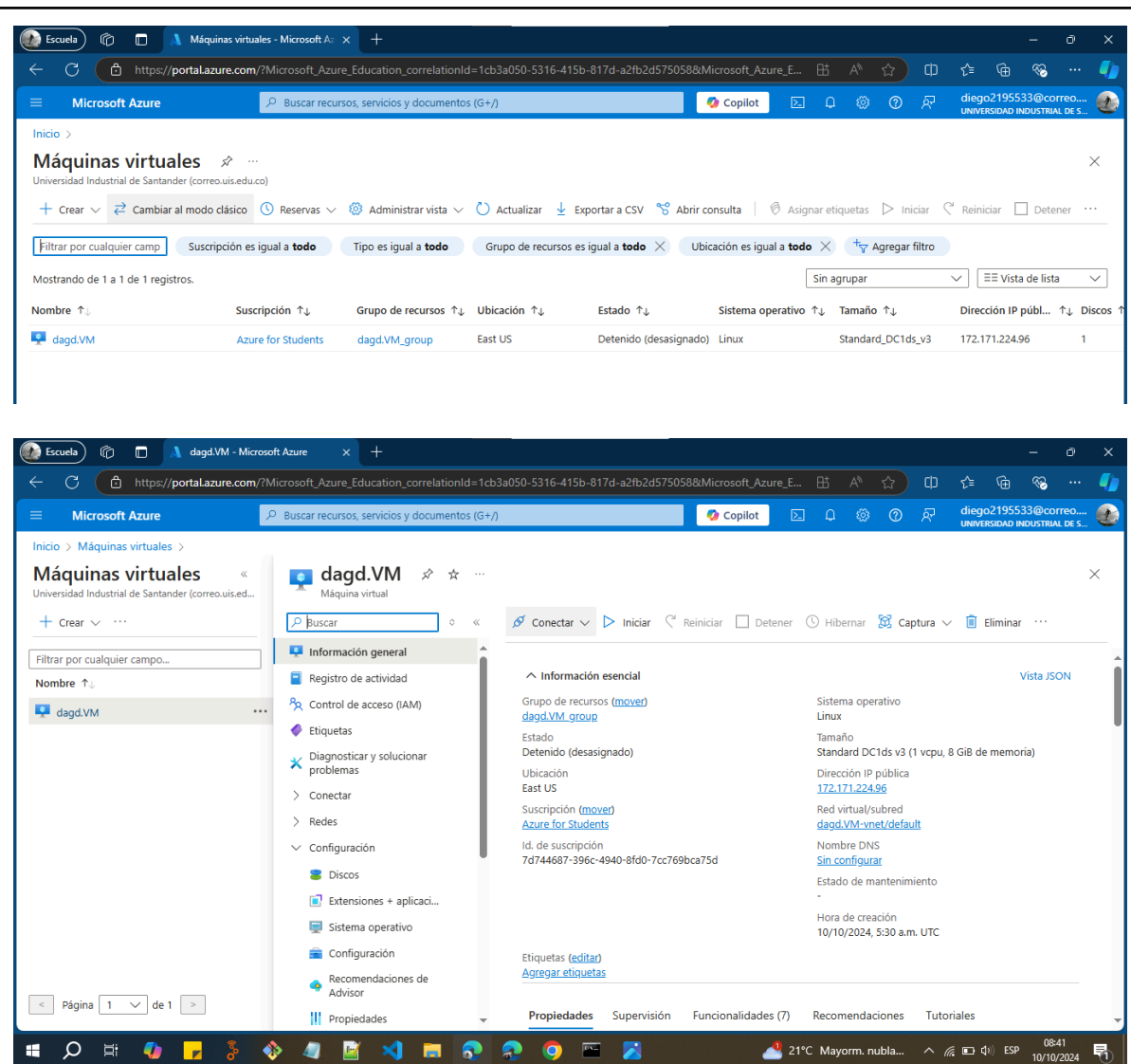
Empaquetar la aplicación y sus dependencias en un contenedor. Esto asegura que la aplicación se ejecute de la misma manera, sin importar el entorno subyacente, ya que el contenedor incluye todo lo necesario para la ejecución.

La portabilidad y consistencia de los contenedores facilitan la migración entre diferentes entornos y proveedores de nube, reduciendo el riesgo de errores y tiempo de inactividad.

# Informe a presentar

A continuación se presentan las tablas que debes llenar para ser presentadas como informe de este taller. Por ahora no debes llenarlas, solo saber que existen. Lo que vas a hacer es pasar a desarrollar las tareas que hay más abajo. En el transcurso de la ejecución de esas tareas, volverás a estas tablas para ir llenándolas.

<b>Título del taller: Registro en Azure y creación de una VM para Mosquitto</b>
<b>Datos de los estudiantes</b>
José David Flores - 2174241 - Ingeniería Electrónica Diego Andrés García - 2195533 - Ingeniería Electrónica
<b>Creación de una VM (Tarea 1 y 2)</b> Explique qué VM ha creado, ¿qué características tiene?. Complementariamente agregue capturas de pantalla sobre ese trabajo



Es una VM con suscripción de estudiantes, sistema operativo Linux, una IP pública y 1 Disco, la VM tiene como nombre **dagd.VM**. La demás información se encuentra en la segunda imagen.

### Acceso remoto a la VM (Tarea 2)

Explique si es posible acceder remotamente, desde un computador Windows, a la VM creada. Para su caso, cuál es el procedimiento a seguir?. Complementariamente agregue capturas de pantalla sobre ese trabajo.

### Procedimiento para acceder remotamente a la VM desde Windows

1. **Obtener la IP pública y credenciales de la VM:**
  - Ir a la página de la máquina virtual en el portal de Azure.
  - Copiar la dirección IP pública y asegúrate de tener el archivo de clave privada (.pem) que se descargó cuando creaste la VM.
2. **Abrir la terminal en Windows (CMD o PowerShell):**
  - Abre el símbolo del sistema (CMD) o PowerShell y dirigirse a la ruta donde se descargó el archivo .pem.
3. **Comando SSH para conectarse a la VM:**



- Usa el siguiente comando para iniciar la conexión SSH:

```
ssh nombre_usuario@dirección_IP -i ruta_al_archivo_pem
```

- Sustituye **nombre\_usuario** por el nombre de usuario que configuraste para la VM.
- Sustituye **dirección\_IP** por la IP pública de la VM.
- Sustituye **ruta\_al\_archivo\_pem** por la ruta completa donde se encuentra el archivo .pem en tu sistema.

#### 4. Aceptación de la autenticidad del host:

- La primera vez que se conecta, es posible que se reciba una advertencia sobre la autenticidad del host. Escribir **yes** para continuar.

#### 5. Acceso exitoso:

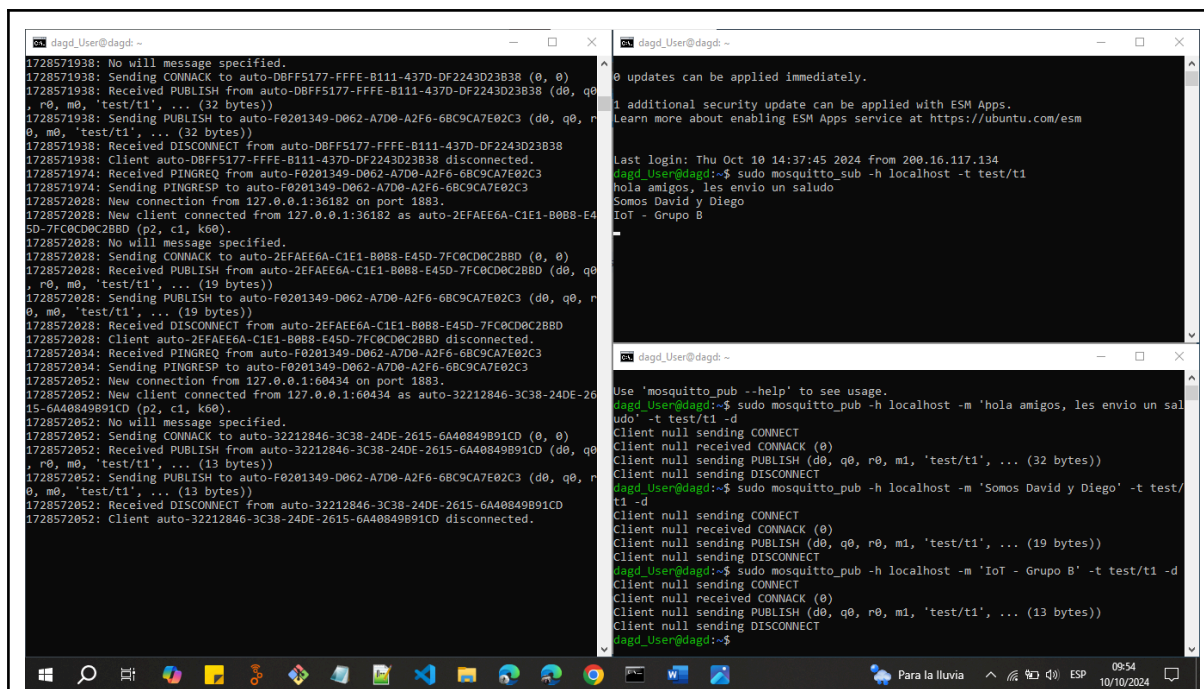
- Si todo está configurado correctamente, se debería estar conectado a la VM en Azure desde la terminal de Windows.

Estado	Sistema operativo	Tamaño	Dirección IP públ...	Disc
En ejecución	Linux	Standard_DC1ds_v3	172.171.224.96	1

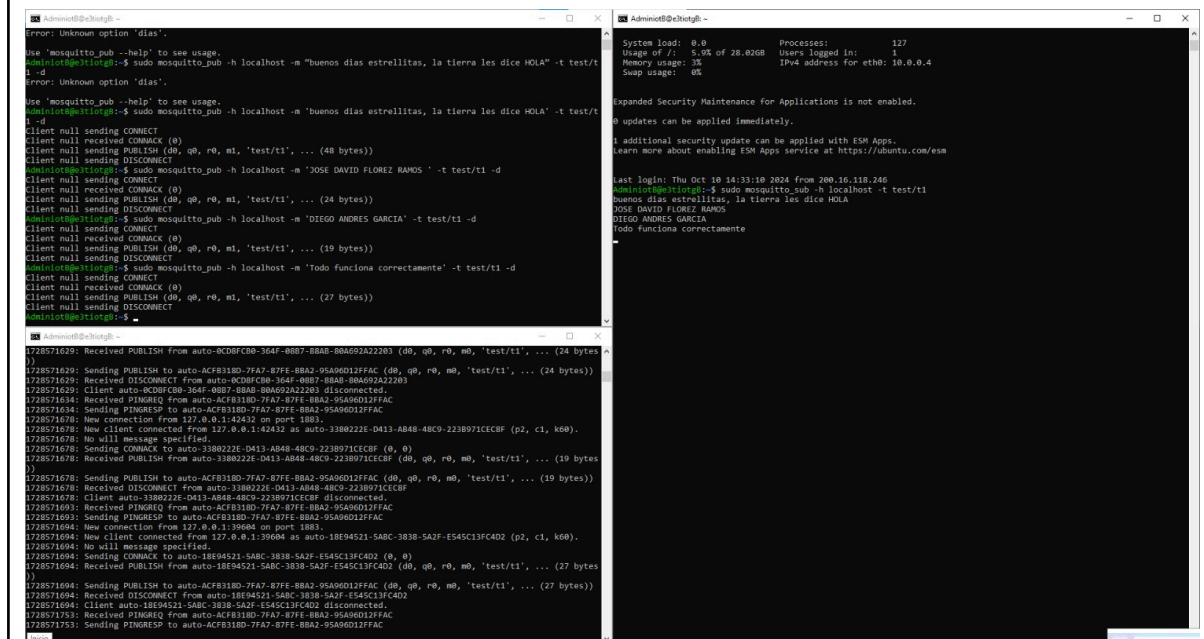
Acá se logra evidenciar la correcta conexión a la VM creada en Azure por medio de CMD (Consola o Terminal) en **Windows 10**, accediendo desde CMD con el comando **ssh dagd\_User@172.171.224.96 -i dagd\_VM\_key.pem** y verificando que la ruta que sale en CMD sea la ruta en donde se encuentra almacenado el archivo **.pem** que se descargo una vez se creó la VM en Azure.

### Instalación y pruebas de Mosquitto (Tarea 3)

Explique el tipo de pruebas llevadas a cabo para comprobar que Mosquitto quedó bien instalado. Complementariamente agregue capturas de pantalla sobre ese trabajo



En la ventana de CMD que está en la parte izquierda se observa el **broker**, en la parte superior derecha, se observa el **suscriptor** y en la parte inferior derecha, se observa el **publicador**. Finalmente se procede a detener la VM desde la página web de Azure.



Acá se muestra la evidencia de lo realizado por mi compañero David. En la parte superior izquierda se observa el **publicador**, en la parte inferior izquierda se observa el **broker**, finalmente en la parte derecha se observa el **suscriptor**.

## Tarea 1. Inscripción a Azure

Siga el [Manual de Azure](#), los capítulos 2 y 3.

## Tarea 2. Crear su primera VM y prueba remotamente usando SSH

Cree su primera VM de Linux siguiendo el [Manual de Azure](#), luego llámese usando SSH desde su computador Windows, capítulo 4

## Tarea 3. Instalar Mosquito a la VM

La idea es instalar Mosquitto y comprobar que funciona, para lo cual, en una misma VM, vamos a crear un sistema completo IoT que contenga al menos un broker, un suscriptor y un publicador. Esto lo hacemos solo con el ánimo de comprobar qué es lo que hemos instalado y si funciona al menos estando todo en una VM. Gracias a esto, sabremos que todo está bien instalado para que más adelante podamos aventurarnos en cosas más complejas, fuera de esta VM

**Paso 1.** Instalación de Mosquitto. Lo haremos siguiendo este [vídeo](#) o los comandos que aparecen a continuación:

Comandos	Para qué sirven
<code>sudo apt-get update</code>	Descarga de los repositorios la información más reciente sobre los paquetes disponibles en los repositorios que mantiene la comunidad de Ubuntu para las posibles actualizaciones de Ubuntu a las que pueda haber lugar, como nuevas versiones o actualizaciones de seguridad. Si nos saltamos este comando, una nueva instalación puede usar, en su instalación, componentes que están desactualizados en la VM.
<code>sudo apt-get upgrade</code>	Si bien, el anterior comando solo descarga información, este procede a la actualización de los paquetes instalados en el sistema a sus últimas versiones disponibles. Acepte las opciones que aparezcan por defecto.
<code>sudo apt install mosquitto mosquitto-clients</code>	<code>apt</code> significa "Advanced Package Tool". Cuando se usa un comando que comienza con <code>apt</code> , se está invocando una herramienta en Ubuntu para gestionar paquetes de software que se encuentran en repositorios. En este sentido, es similar a GitHub, pero mientras GitHub gestiona repositorios de código fuente, <code>apt</code> gestiona paquetes binarios de software en servidores

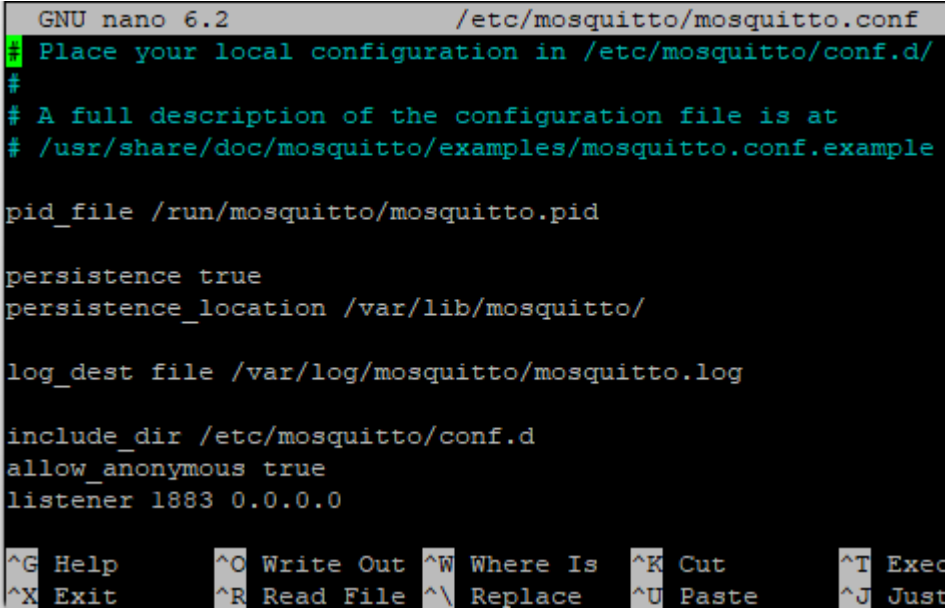
	<p>dedicados a Linux. Se dice que es un comando administrativo y como tal debe ser ejecutado como superusuario anteponiendo sudo.</p> <p>apt-get es se refiere a órdenes más antiguas que las que comienzan con apt. Conllevan a lo mismo, pero apt da resultados visuales más amigables aunque menos detalladas</p> <p>Por eso, esta orden conduce a la instalación de aquella versión de Mosquitto que está incluida en la versión de Ubuntu que estás utilizando. En otras palabras, pueden existir versiones más avanzadas del paquete, pero la que se instala es la que ha sido re-comprobada para su versión de Ubuntu.</p> <p>En este caso se instala mosquitto &gt; a las preguntas que surjan acepte las opciones que aparecen por defecto</p> <p>Nota 1: Este comando representa dos comandos en uno solo. Por eso, otra opción sería enviar los siguientes dos comandos:  sudo apt-get install mosquitto  sudo apt-get install mosquitto-clients</p> <p>Nota 2: el parámetro “mosquitto-clients” es una orden para instalar no solo el broker mosquito, sino también los clientes. Entonces, si vamos a usar nuestra máquina en algún momento como un cliente (suscriptor o publicador) es necesario incluir esta opción. Un desarrollador querrá tener clientes para poder hacer pruebas más diversas. Un servidor en producción, para que sea muy liviano, no necesita clientes, porque ellos pueden ser externos.</p> <p>Nota 3: si uno quiere la última versión tomada directamente de los repositorios de Mosquito, se puede actualizar previamente el repositorio de Mosquito en tu sistema con:  <pre>sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa sudo apt-get update</pre></p>
sudo apt-get install net-tools	Permite instalar herramientas en Ubuntu que serán útiles para conocer estados de

	conectividad de la VM. Hay herramientas como: ifconfig, netstat, route, arp, nameif, dnsdomainname > a las preguntas que surjan acepte las opciones que aparecen por defecto
sudo systemctl enable mosquitto.service	Systemd es el sistema de administración de servicios de Ubuntu. systemctl es un subcomando de systemd. Con comandos Systemctl es posible obligar al sistema a realizar ciertas tareas con los paquetes instalados. En este caso se está configurando el sistema para que el servidor de Mosquitto se inicie automáticamente durante el arranque del sistema.
sudo systemctl daemon-reload	Reinicia el demonio del sistema. Eso significa que si realizaste alguna configuración, esta entrará en acción con este reinicio. Es como una orden para que los cambios se apliquen. Recarga la configuración del demonio del sistema para asegurarse de que los cambios realizados en la configuración o unidad systemd se apliquen correctamente.
sudo service mosquitto restart	Para reiniciar el servidor de Mosquitto y puede ser útil en situaciones en las que se ha realizado alguna modificación en la configuración o si se ha detectado algún problema en el servidor

## Paso 2. Configuración del Mosquitto

- Estas recomendaciones son tomadas de este [video](#)
- Mosquitto tiene una configuración interna por defecto, pero el archivo mosquitto.conf permite que nosotros forcemos ciertos efectos. Por ejemplo, la configuración interna no permite clientes anónimos sino locales, lo cual no nos conviene si queremos que desde la web un cliente se conecte a nuestro servidor de manera anónima.
- abrir el archivo mosquitto.conf, que está en etc/Mosquitto y le agregamos unas líneas, siguiendo estos pasos:

Comandos	Descripción
cd /etc/mosquitto	Nos pasamos a la ubicación del archivo de configuración de mosquitto
sudo cp mosquitto.conf copia.mosquitto.conf	Esto es para crear una copia del archivo original por si acaso necesitamos recuperarlo

sudo nano mosquitto.conf	se corre el editor de texto llamado nano y este a su vez abre el archivo mosquitto.conf
allow_anonymous true listener 1883 0.0.0.0	<p>Son las líneas que se agregan al final del archivo mosquitto.conf. Con esto, se asegura que:</p> <ul style="list-style-type: none"> <li>• allow_anonymous true. Sirve para que los clientes MQTT se puedan conectar al servidor de Mosquitto sin proporcionar credenciales de autenticación, como usuario y contraseña.</li> <li>• listener 1883 0.0.0.0. Hace que el Servidor Mosquitto se quede escuchando el puerto 1883 en todas las interfaces de red disponibles en el sistema. El puerto 1883 es el puerto predeterminado para la comunicación MQTT y está diseñado para ser utilizado para conexiones no encriptadas. "0.0.0.0" indica que el servidor de Mosquitto aceptará conexiones MQTT entrantes desde cualquier dirección IP</li> </ul>
log_dest stdout	La configuración por defecto de mosquitto no entrega un log en la terminal. Con esta línea si se logra.
	
Ctrl+O aceptar > Ctrl+X	Sirve para que los cambios hechos al archivos se guarden y para salir de nano
Reiniciamos la VM para que apliquen los cambios realizados a la configuración	

### Paso 3. Pruebas de funcionamiento de mosquitto

Abre dos terminales adicionales de powershell y logeate en cada una a la VM, así:

La primera terminal, la que ya tenemos es el broker

La segunda a la izquierda es el suscriptor

La tercera a la derecha es el publicador

Siga este [Videotutorial para la prueba](#) o bien los comandos de la siguiente tabla:

Comandos	Explicación
<code>sudo systemctl stop mosquitto</code> <code>sudo mosquitto -v</code>	En la terminal de broker detenemos mosquitto para poder re-abrirlo pero en modo verborreico, o sea que mosquitto queda en suspenso, como cuando uno da el comando python.
<code>sudo mosquitto_sub -h localhost -t test/t1</code>	En la terminal de suscriptor te suscribes al tópico "test/t1" mediante este comando. Mira la reacción en el broker y en el mismo suscriptor
<code>sudo mosquitto_pub -h localhost -m "hola amigos, les envio un saludo" -t test/t1 -d</code>	En la terminal de publicador creas el tópico "test/t1" y envías un mensaje mediante este comando. Mira la reacción en el broker, en el suscriptor y en el mismo publicador

Otros comandos que pueden llegar a ser útiles en ciertos momentos de uso cotidiano

Comando	Descripción
<code>dpkg -l   grep mosquitto</code>	Sirve para verificar si mosquitto está instalado
<code>sudo systemctl status mosquitto</code>	Para saber si mosquitto está corriendo
<code>ps -ef   grep mosquitto</code>	<p>El comando <code>ps -ef   grep mosquitto</code> se utiliza para buscar información sobre los procesos relacionados con "mosquitto" que están actualmente en ejecución en tu sistema. Y es que uno de los errores comunes es cuando nosotros, haciendo pruebas terminamos iniciando una nueva instancia de mosquitto, pero dos instancias a la vez no pueden usar los mismos puertos. El comando ayuda a ver si hay más de una instancia abierta. Vamos a desglosar lo que hace cada parte del comando:</p> <p>Desglose del comando <code>ps -ef</code>:</p>



	<p>ps: Muestra información sobre los procesos en ejecución.</p> <p>-e: Muestra todos los procesos en el sistema, no solo los asociados al terminal actual.</p> <p>-f: Muestra una lista completa en formato extendido, proporcionando más detalles sobre cada proceso (como UID, PID, PPID, C, STIME, TTY, TIME, CMD).</p> <p>  (pipe):</p> <p>El símbolo de pipe ( ) se utiliza para enviar la salida del comando ps -ef como entrada al siguiente comando, grep.</p> <p>grep mosquito:</p> <p>grep: Un comando de búsqueda que filtra la entrada para mostrar solo las líneas que coinciden con el patrón de búsqueda especificado.</p> <p>mosquitto: El patrón de búsqueda. En este caso, estamos buscando cualquier línea que contenga la palabra "mosquitto".</p> <p>Propósito del comando</p> <p>El comando completo ps -ef   grep mosquito busca y muestra los procesos relacionados con "mosquitto" que están actualmente en ejecución en el sistema. Esto es útil para:</p> <p>Verificar si Mosquitto está corriendo: Puedes confirmar si el broker Mosquitto está en funcionamiento.</p> <p>Obtener el PID: Puedes encontrar el ID de proceso (PID) de Mosquitto, lo cual puede ser útil para tareas de administración como reiniciar o detener el servicio.</p> <p>Diagnóstico: Si estás teniendo problemas con Mosquitto, este comando te ayuda a ver los procesos activos y obtener más detalles sobre ellos</p>
mosquitto -v	<p>Sirve para abrir una instancia de mosquitto en modo verborreico o sea que la terminal queda abierta y dedicada para que mosquitto muestre allí acontecimientos. Pero OJO: antes de usar este comando debes parar mosquitto (con sudo systemctl stop mosquitto) si este ya está corriendo, porque como se dijo antes no pueden haber dos instancias usando los mismos puertos.</p>