# **Manual de Instalación**

## **Sistema MotionParts**

## **Tabla de Contenidos**

1. [Introducción](#_heading=h.ynes77k4c63l)
2. [Requisitos Previos](#_heading=h.8uut0ewislvw)
3. [Preparar Instancia en AWS](#_heading=h.tb5ncf10asgo)
4. [Instalación de Docker en Ubuntu](#_heading=h.giga1bvgq3uo)
5. [Configuración del Proyecto](#_heading=h.geh8hb6kbf59)
6. [Certificado SSL](#_heading=h.gt5prpph91jp)
7. [Acceso y Verificación](#_heading=h.dqvo976a6kwq)

# Introducción

Este manual describe el proceso completo de instalación y configuración del sistema MotionParts en un servidor Ubuntu utilizando Docker como plataforma de contenedores y Caddy como servidor web con SSL automático.

El sistema consta de tres componentes principales:

* **Base de datos:** MongoDB
* **Backend:** API REST en Spring Boot (Puerto 8080)
* **Frontend:** Aplicación Angular (Puerto 4200)
* **Portainer:** Herramienta de monitoreo de contenedores (Puerto 9000)

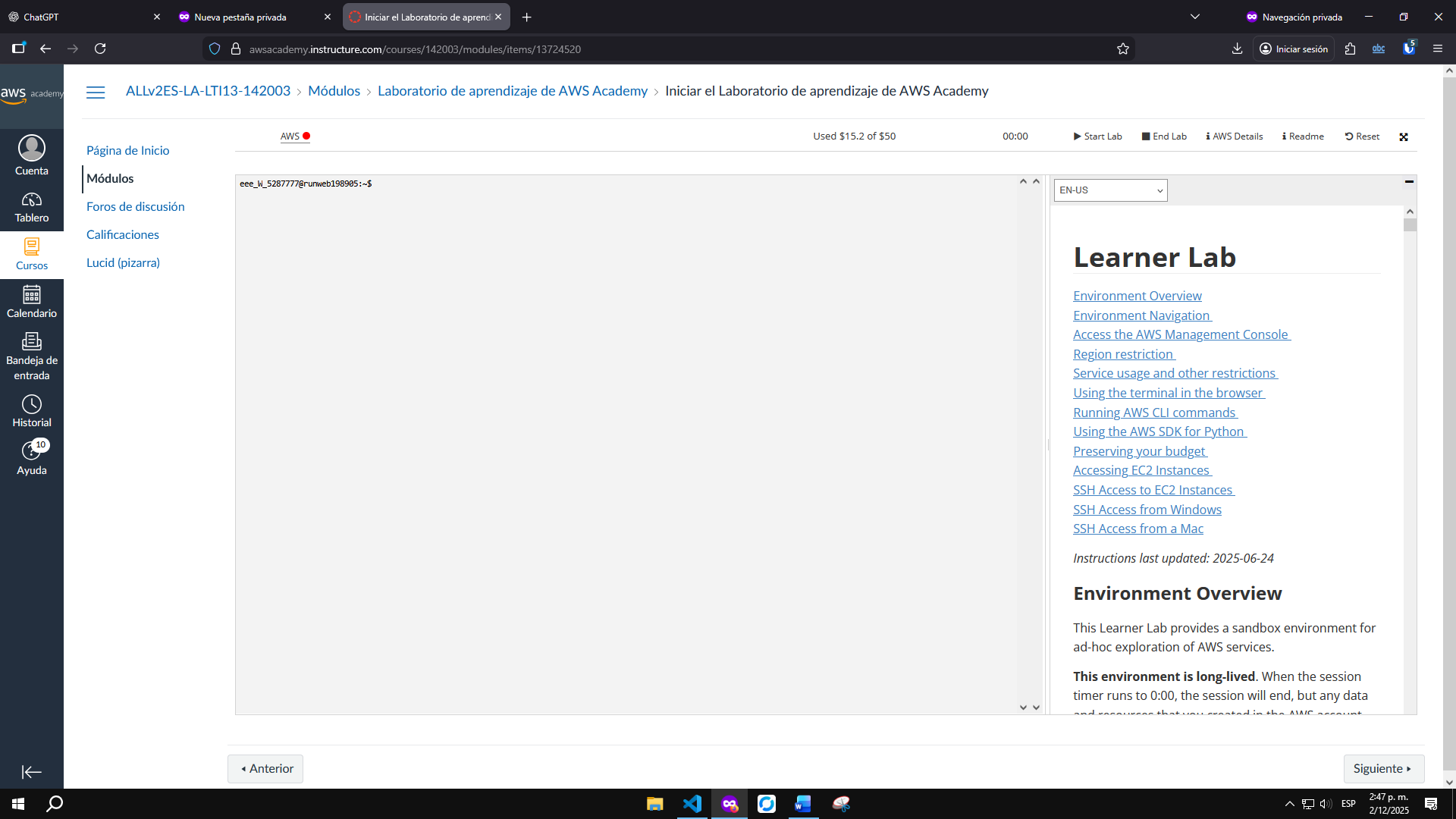
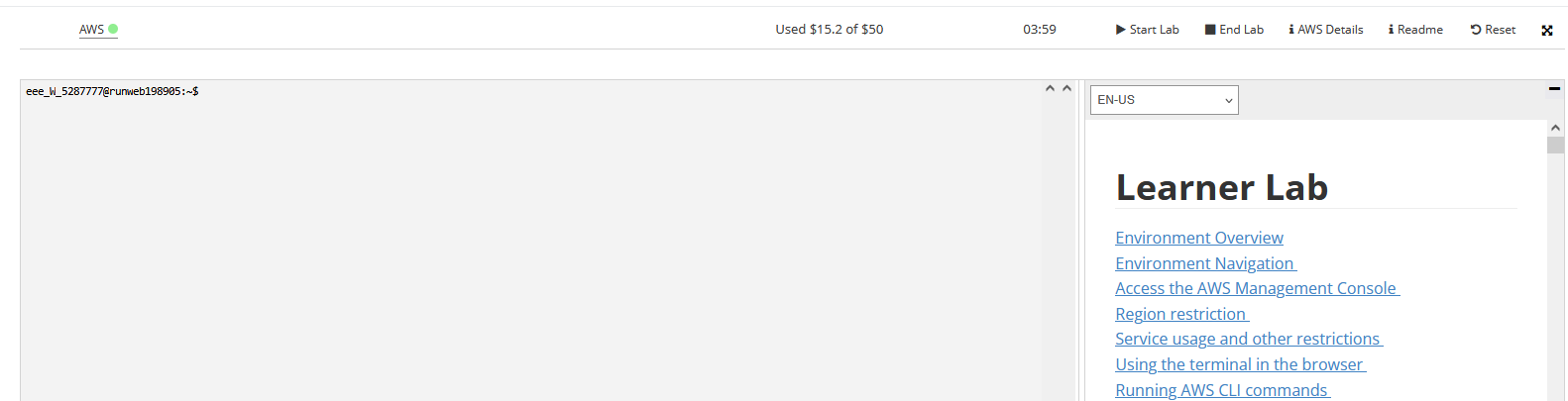
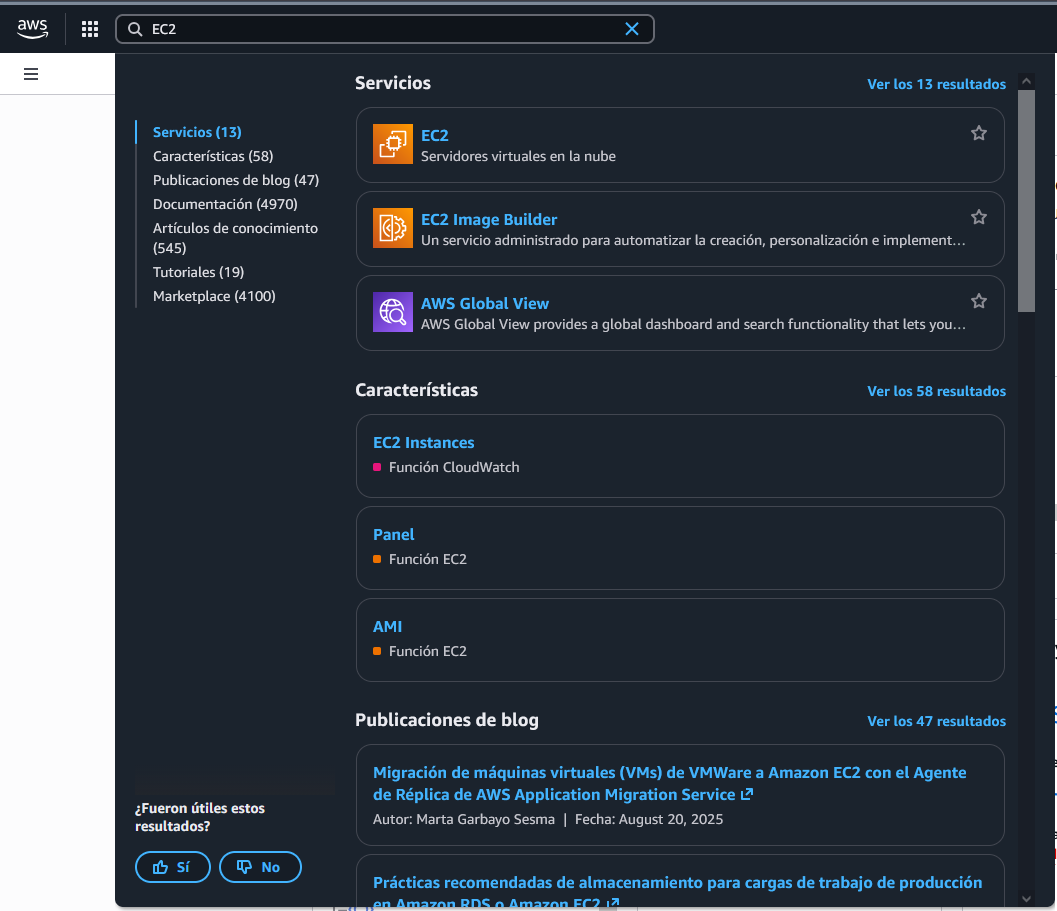
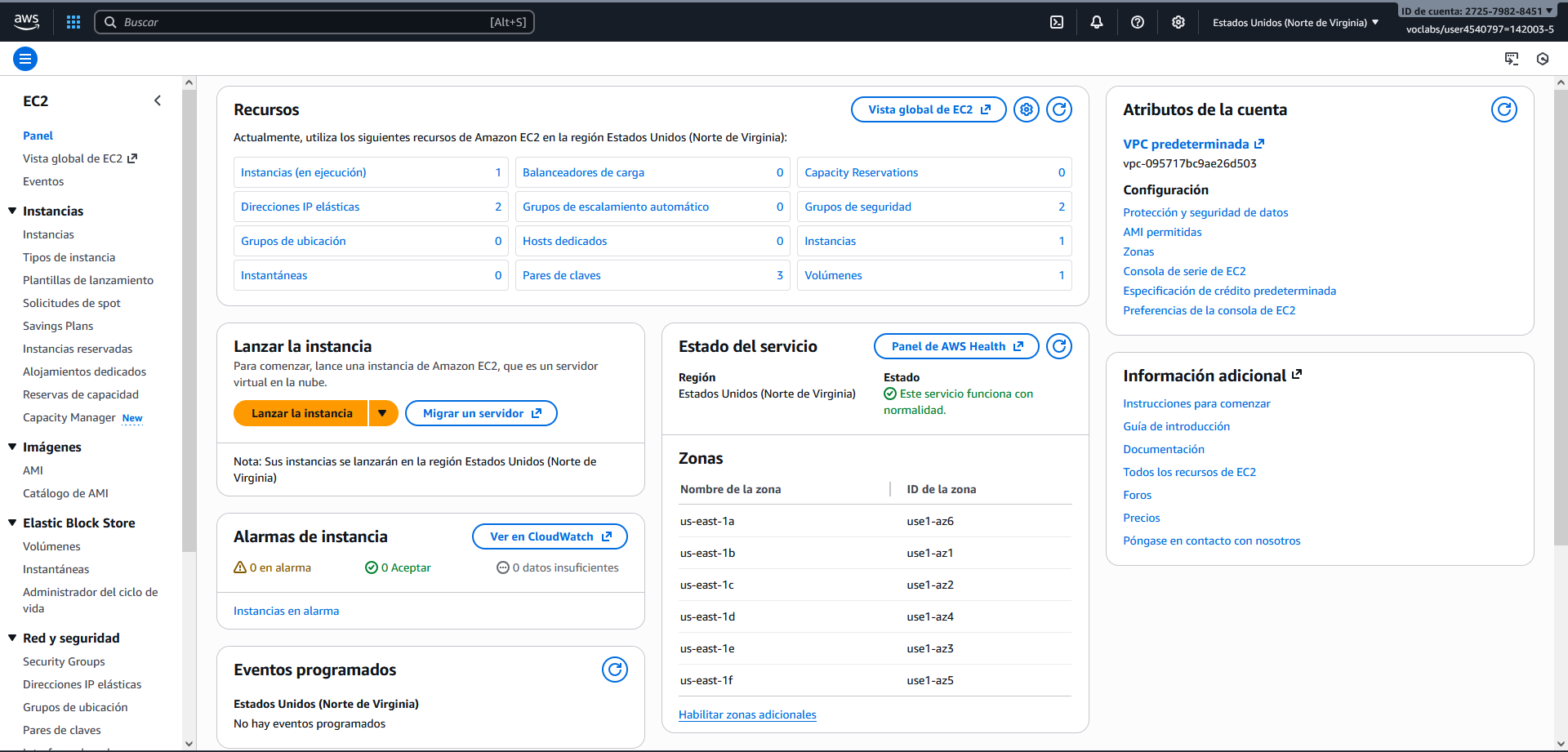
# Requisitos

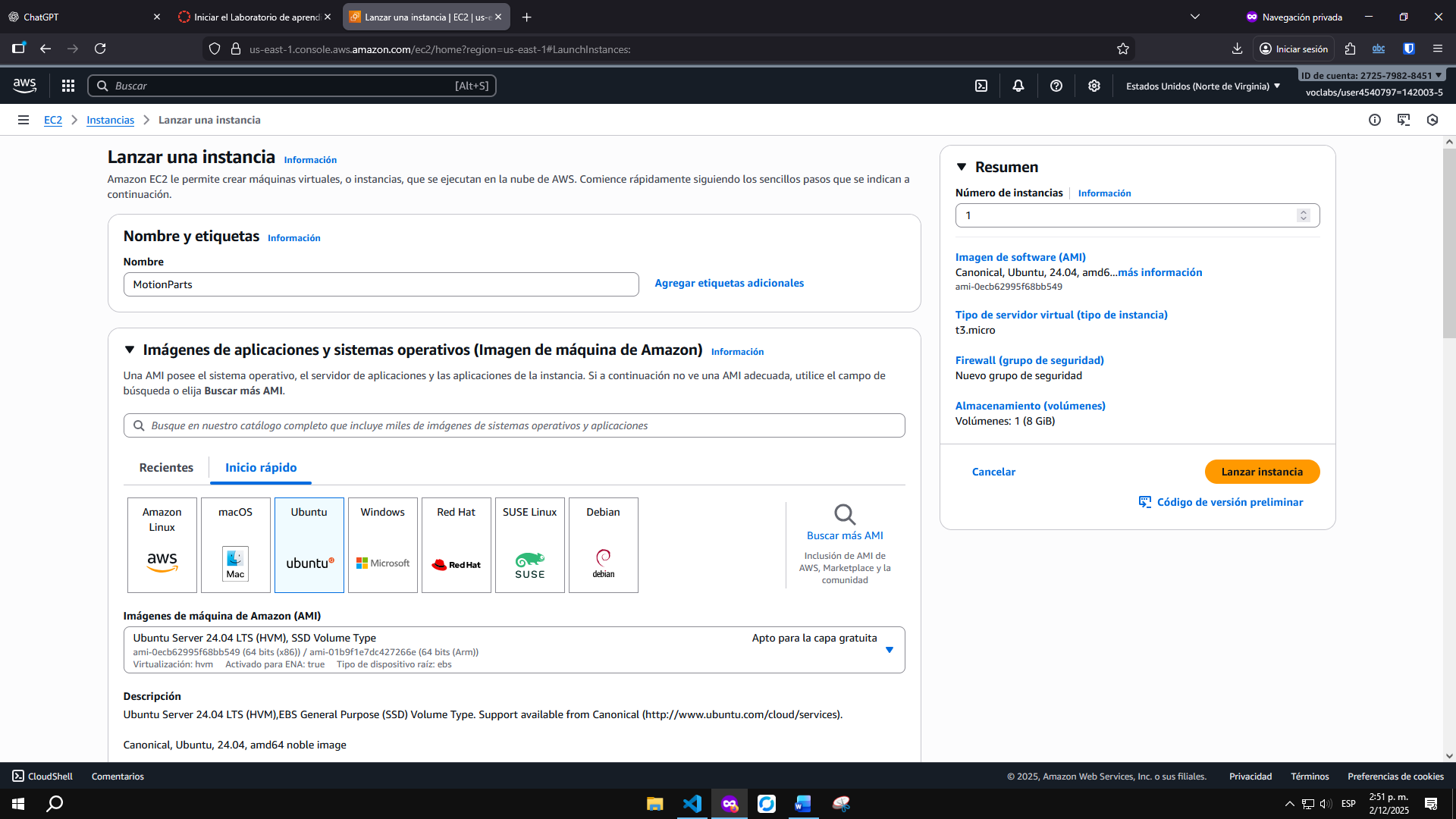
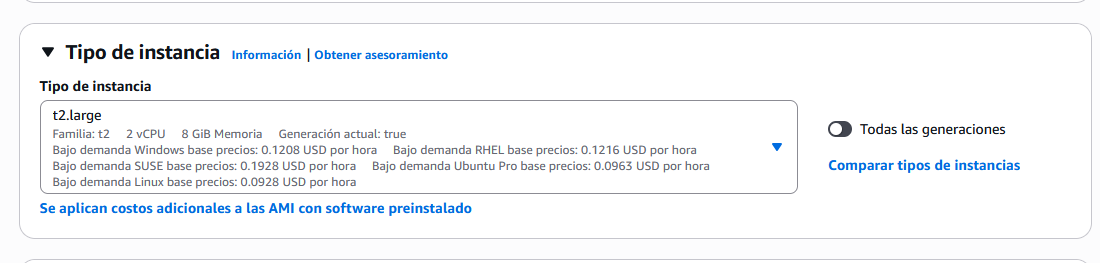
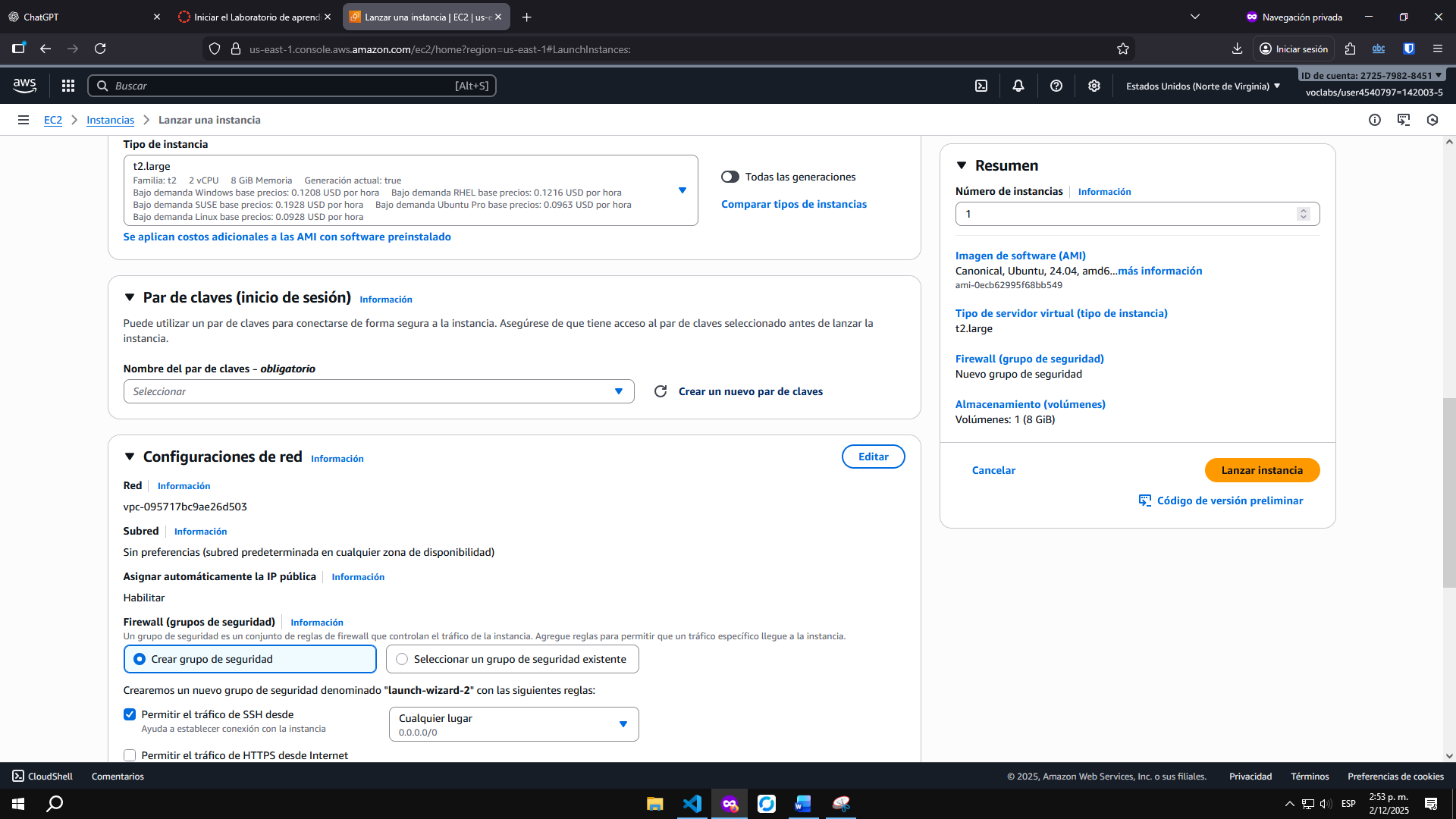
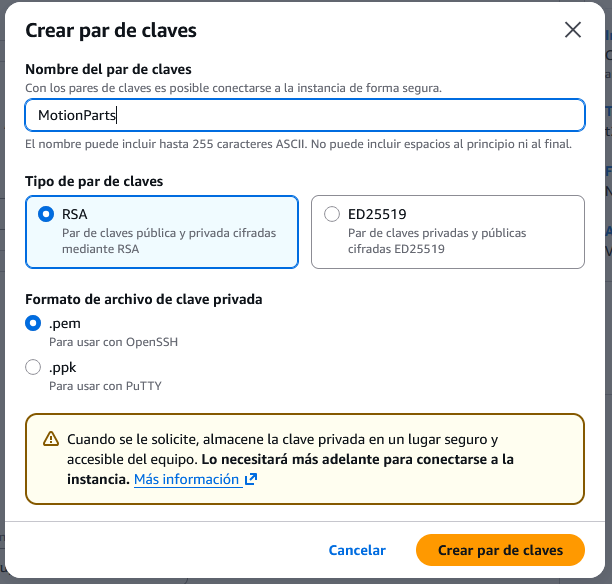
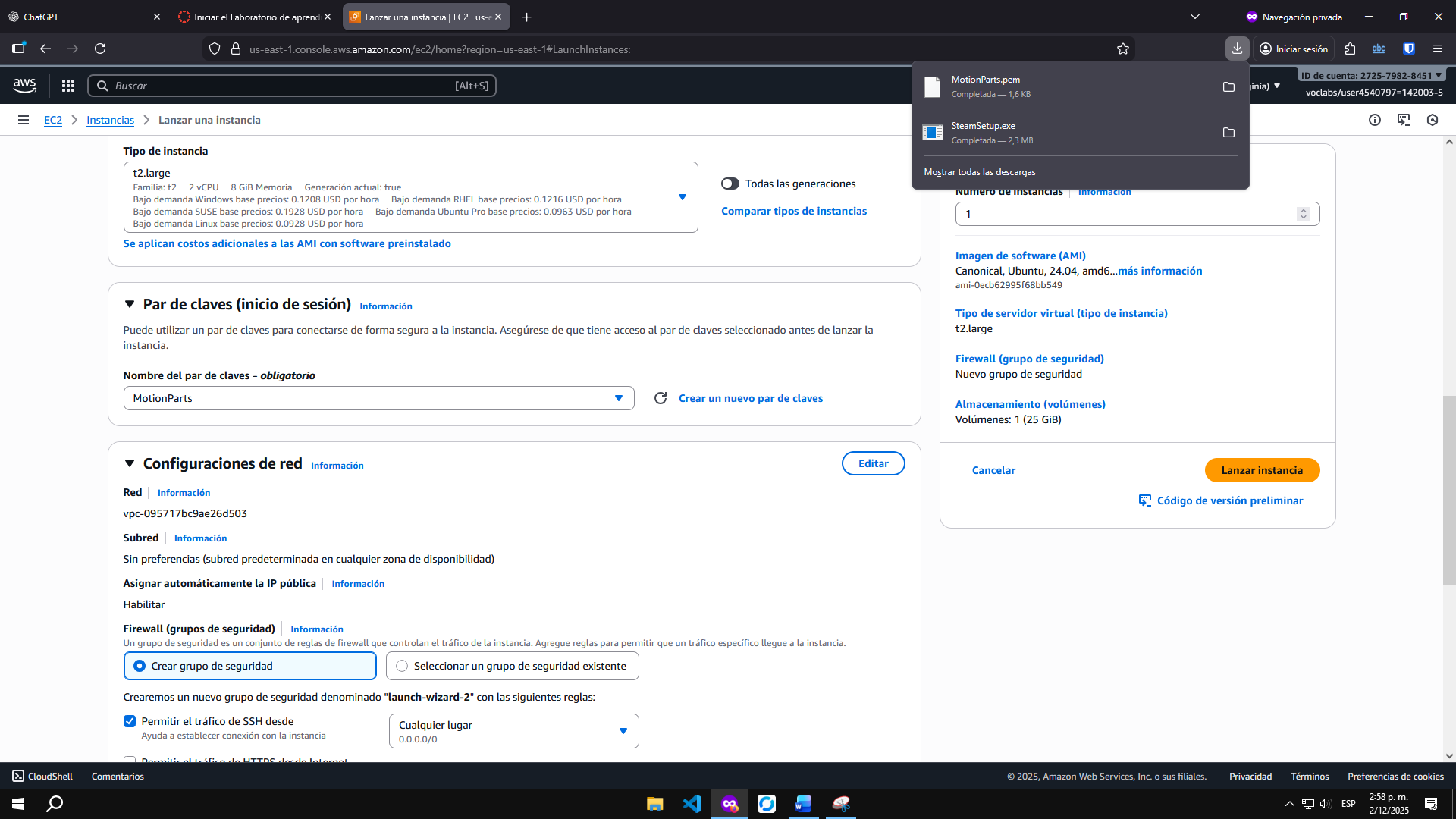
Antes de comenzar la instalación, asegúrese de contar con:

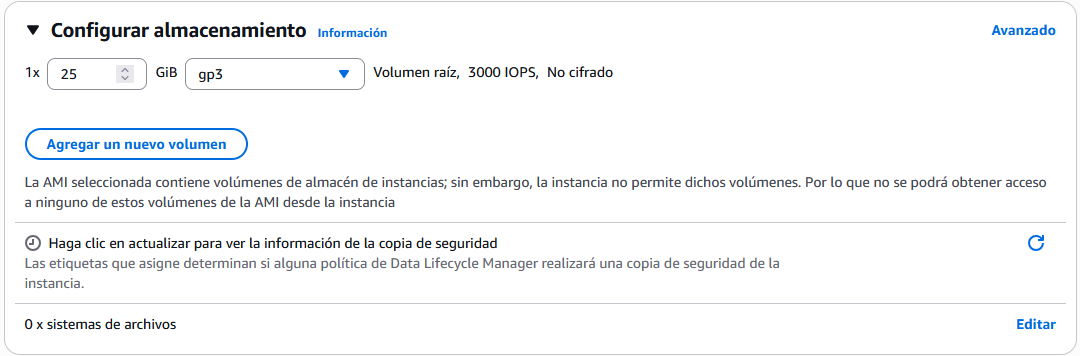
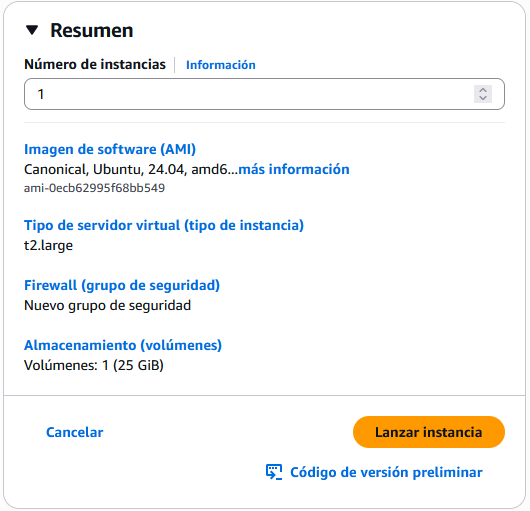
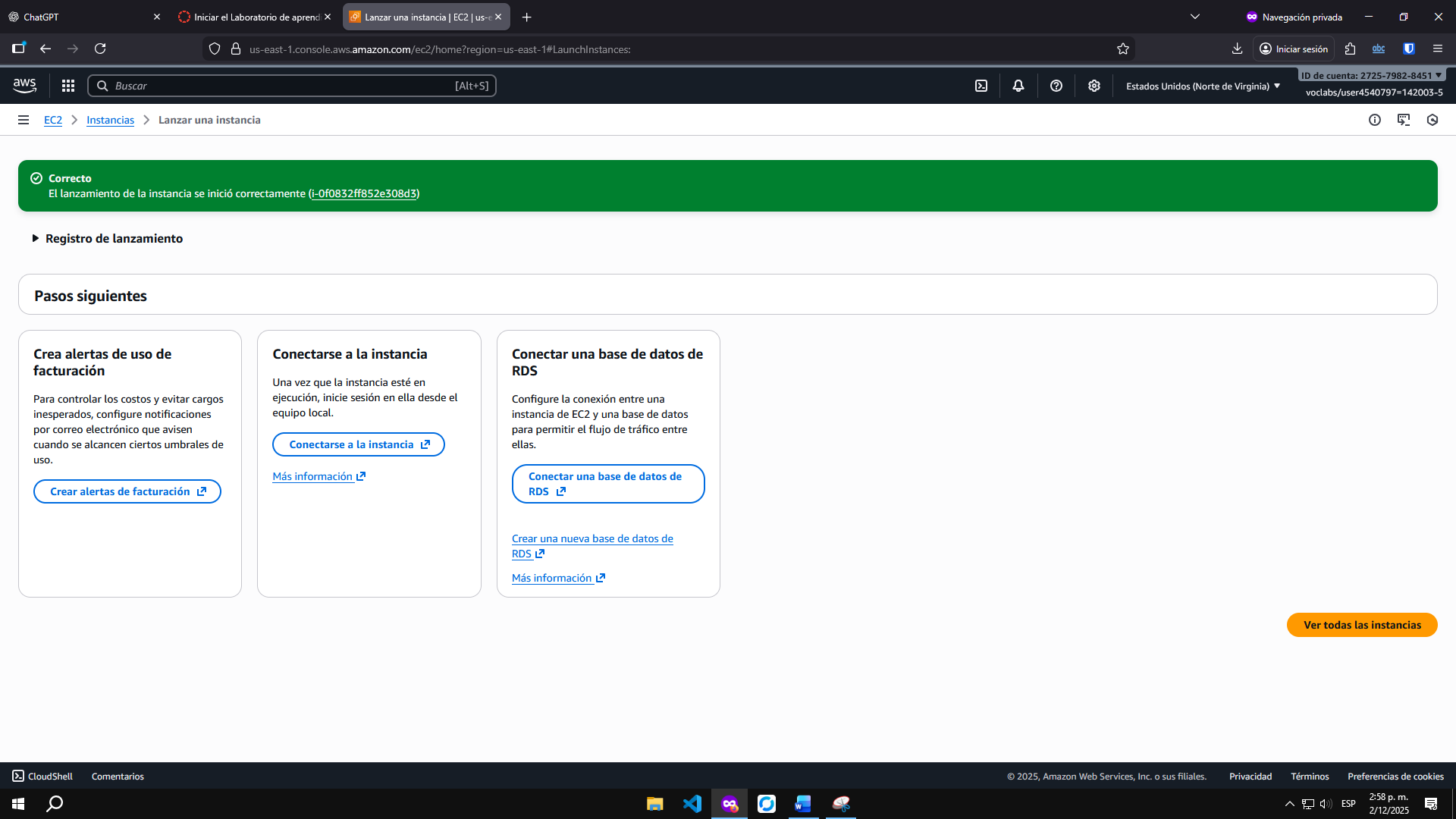
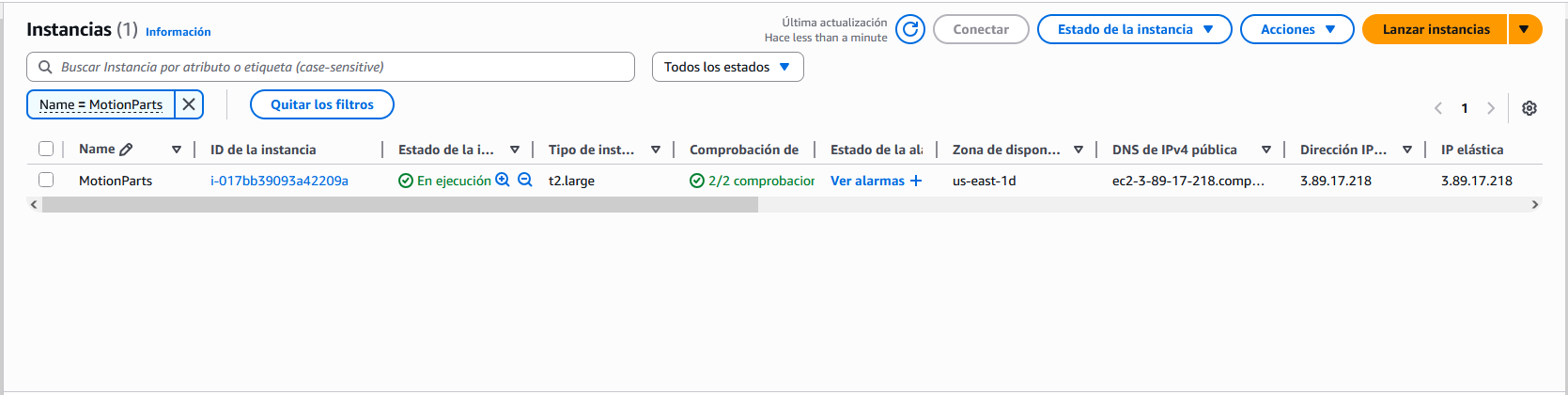
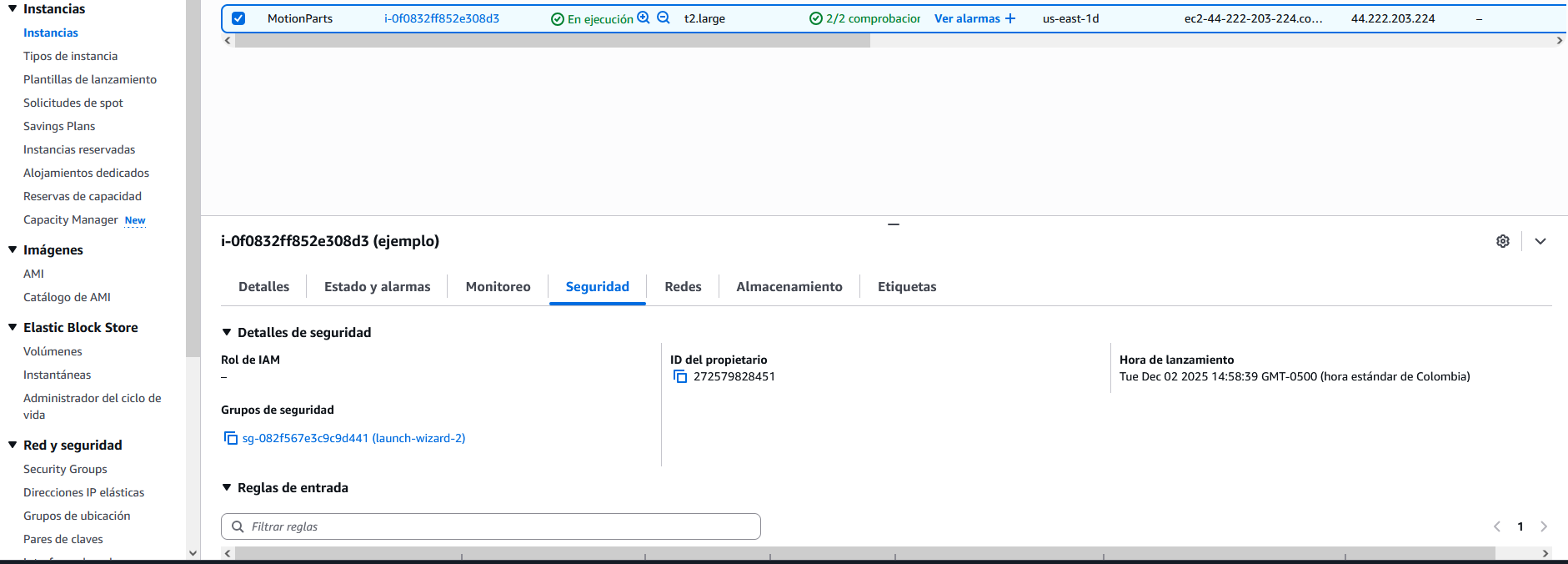
* Servidor Ubuntu 20.04 o superior
* Acceso root o usuario con privilegios sudo
* Conexión a Internet estable
* Dominio configurado (por ejemplo: dominio.org o servicio DuckDNS)
* Puertos abiertos: 80, 443, 4200, 8080, 9000

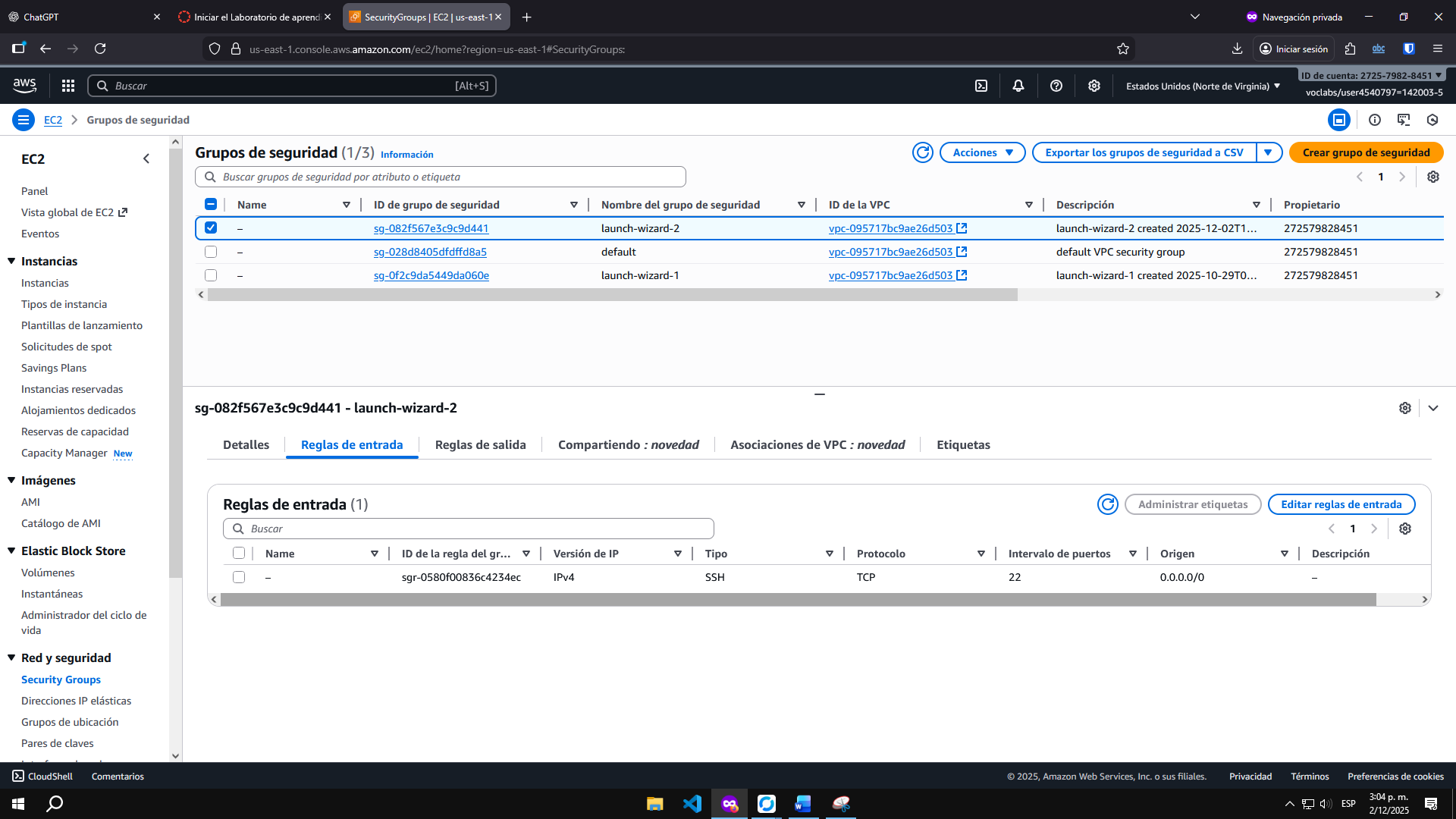
# **Preparar Instancia en Canvas - AWS** **1. Iniciar sesión en Canvas** 1. Inicie sesión en Canvas con las credenciales otorgadas y dirígete al aula, te debe aparecer esta pantalla a continuación:

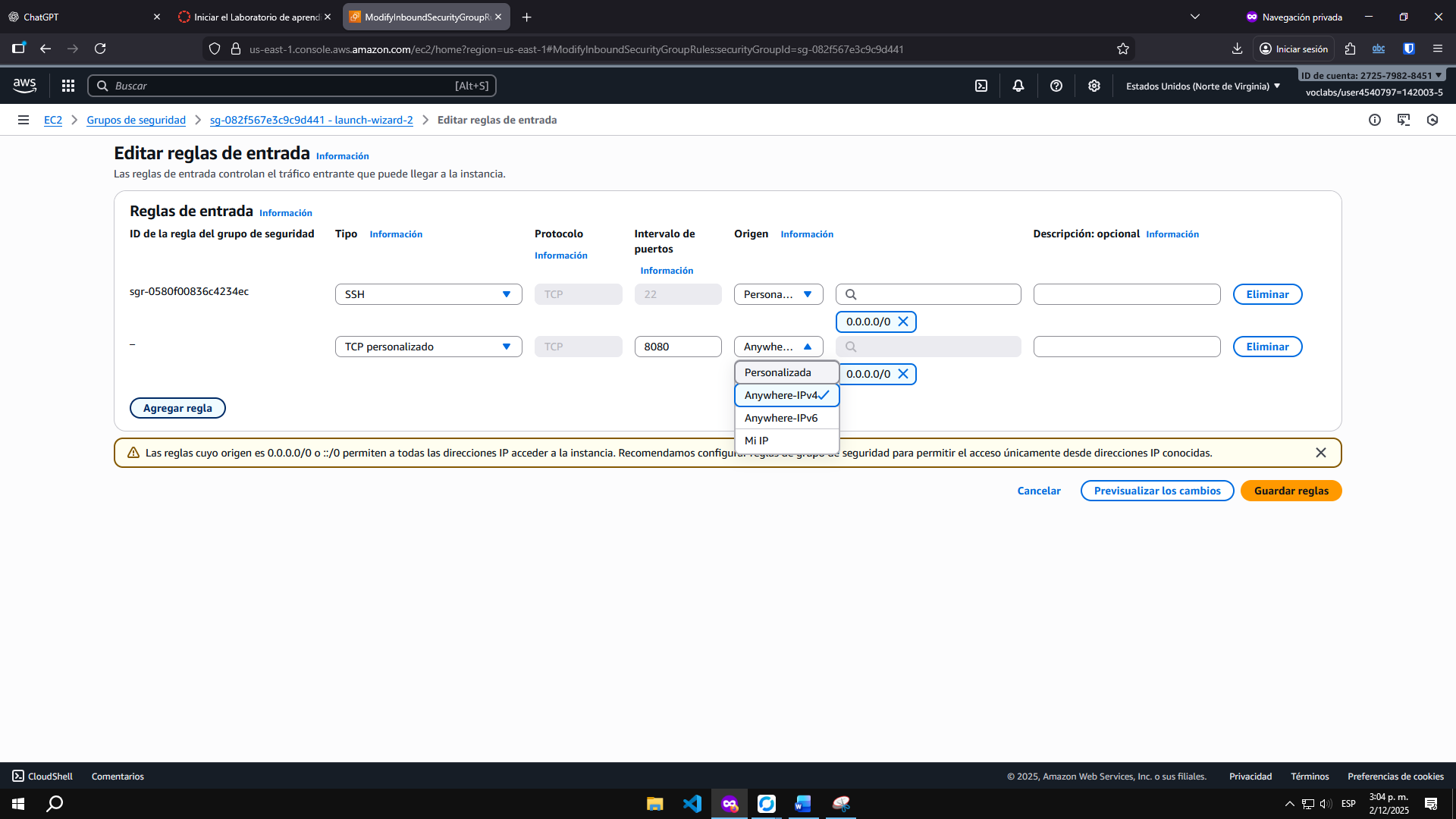
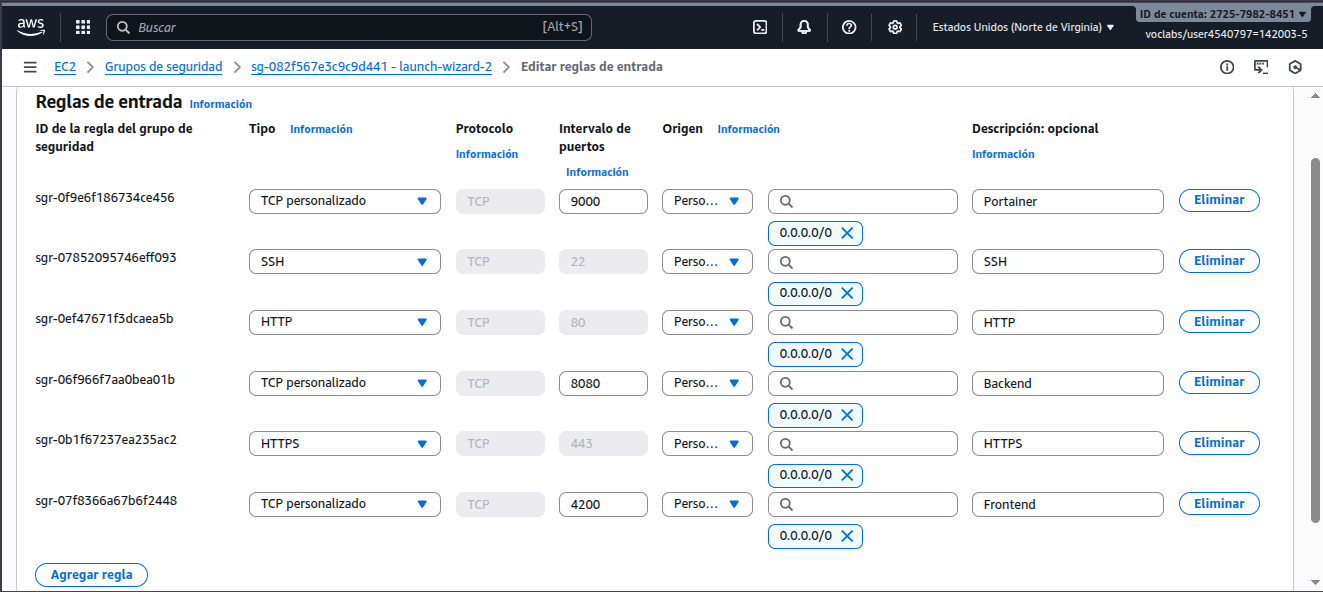
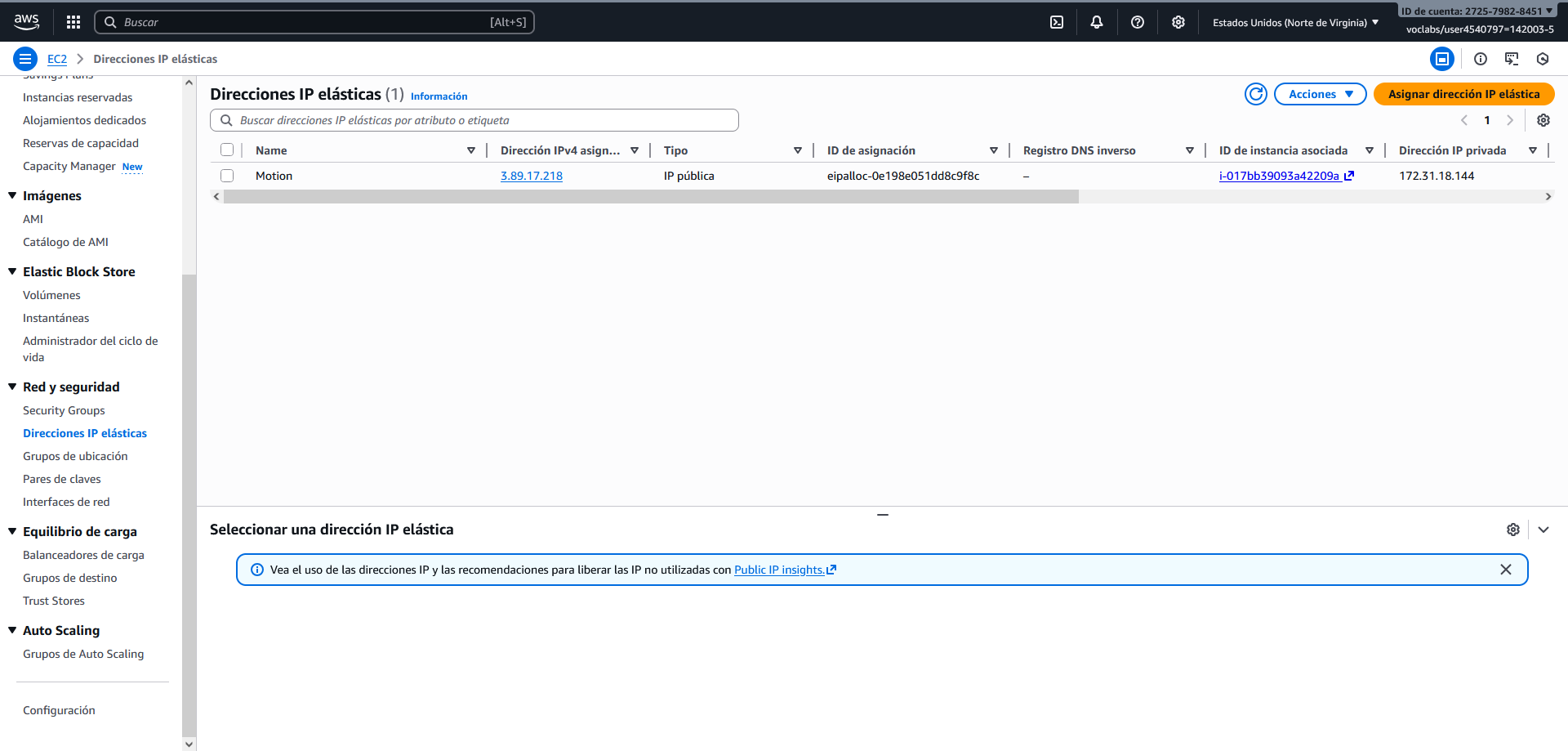
2. Seleccione la sección de módulos y dale click a “Laboratorio de aprendizaje de AWS academy”  
  
  
3. Dentro del laboratorio, debes darle click “Start lab”

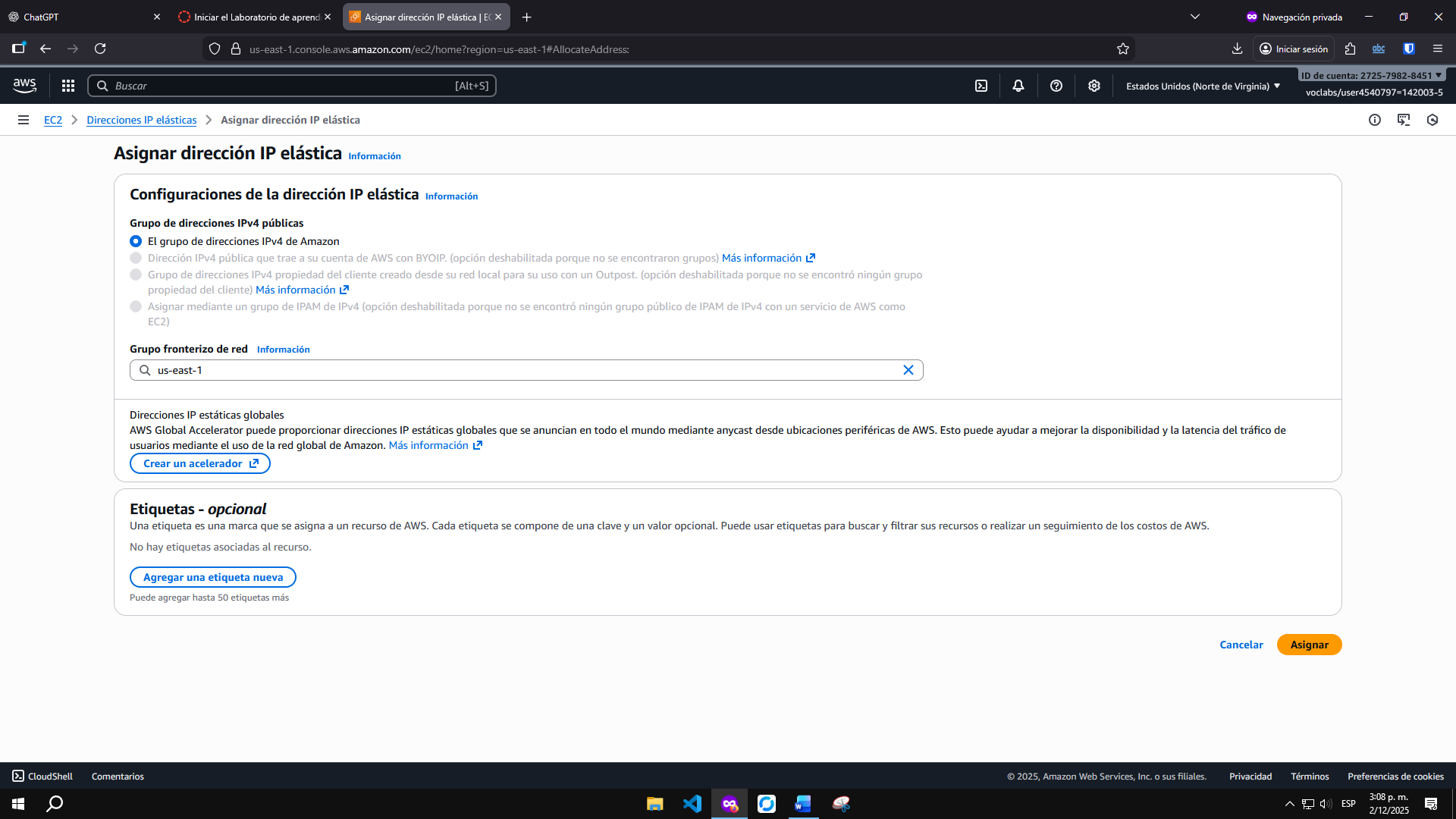
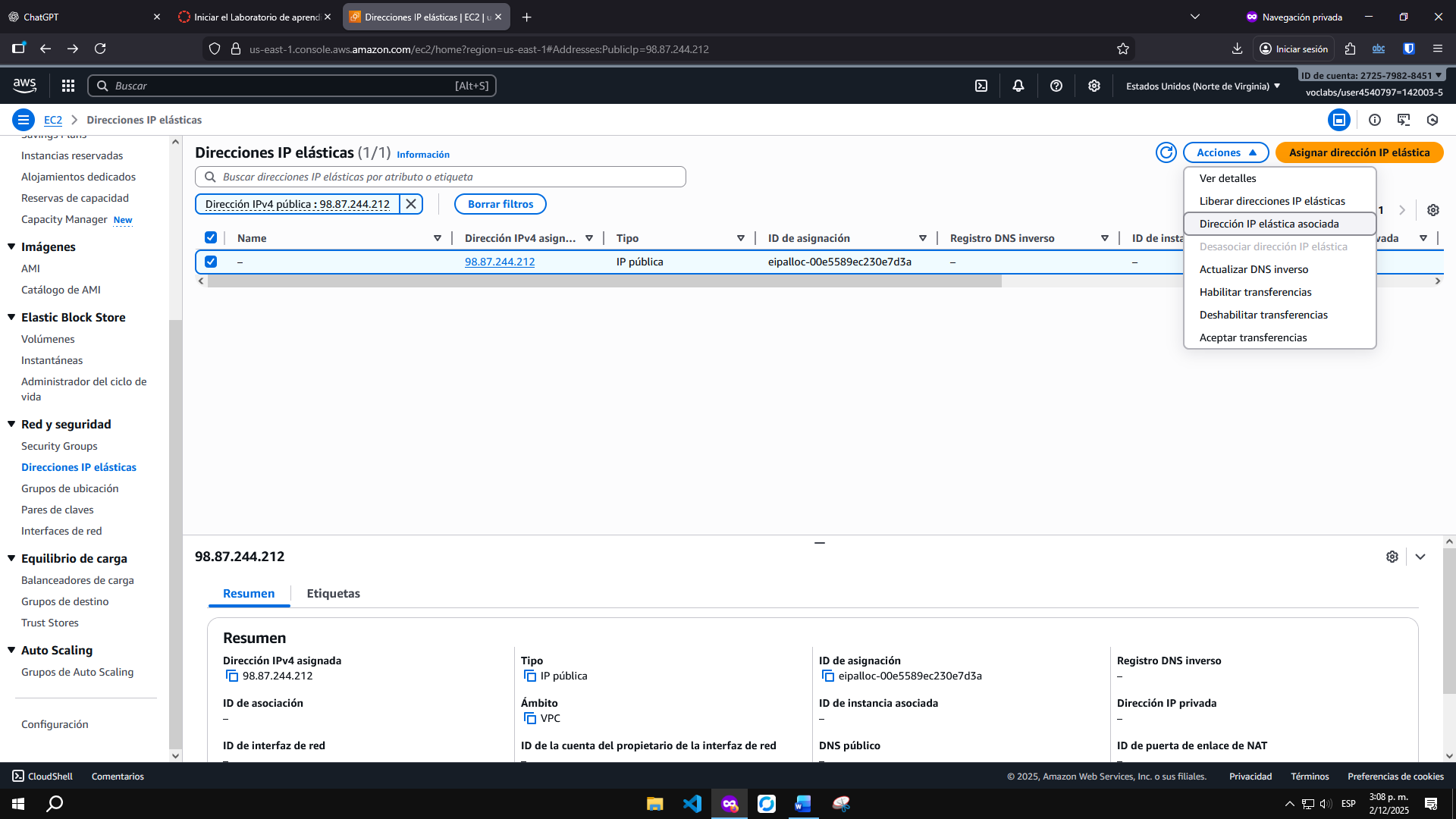
  
  
El boton de color rojo “AWS” debe cambiar a color verde, el laboratorio ya esta activo, a continuacion, haga click en el boton verde “AWS”, esto lo redireccionara a AWS  
  
  
  
  
  
  
4. Dentro de AWS, en el buscador debe buscar el servicio “EC2”  
  
  
Al darle click, aparecerá esta pantalla a continuación:  
  
En la sección “Lanzar la instancia”, dar click al botón naranja “Lanzar la instancia”  
  
  
  
**2. Configuración de la instancia**  
1. Asignar un nombre a la instancia, en la seccion “Imagenes de aplicaciones y sistemas operativos (imagen de maquina de Amazon)” seleccionar Ubuntu

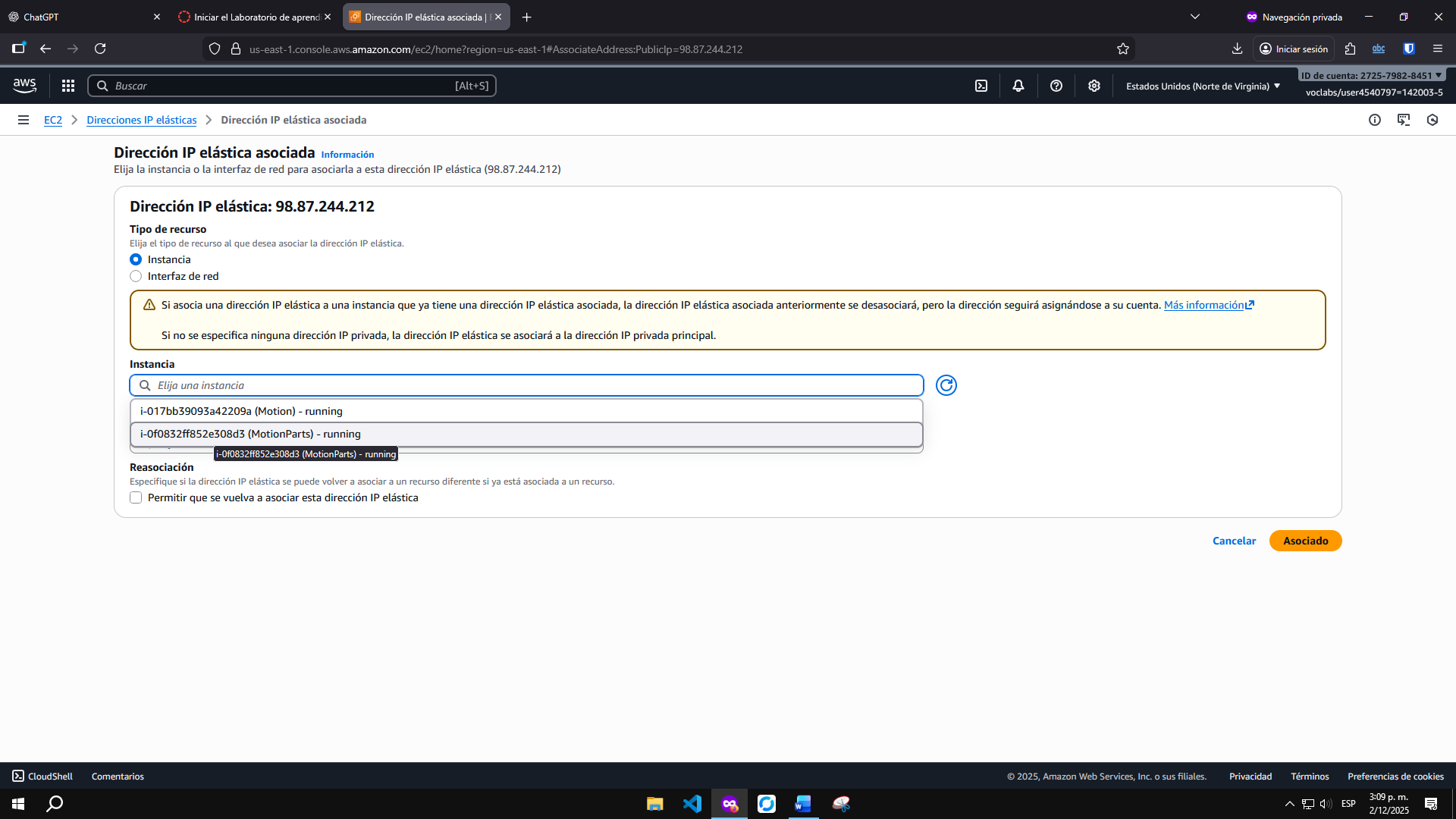
  
  
Más abajo se encontrará con la sección “Tipo de instancia”, se debe seleccionar “t2.large”  
  
  
En la sección “Par de claves”, se deben crear un nuevo par de claves en el botón “Crear un nuevo par de claves”, es importante para poder acceder a la máquina  
  
  
se asigna un nombre con la siguiente configuración:  
  
  
Esto descargara un archivo .pem, este es nuestro par de claves  
  
**Nota**: Resguardar y proteger muy bien el archivo, en caso de pérdida o eliminaciòn del archivo, va a ser imposible acceder a la instancia y es necesario crear una nueva instancia con claves nuevas

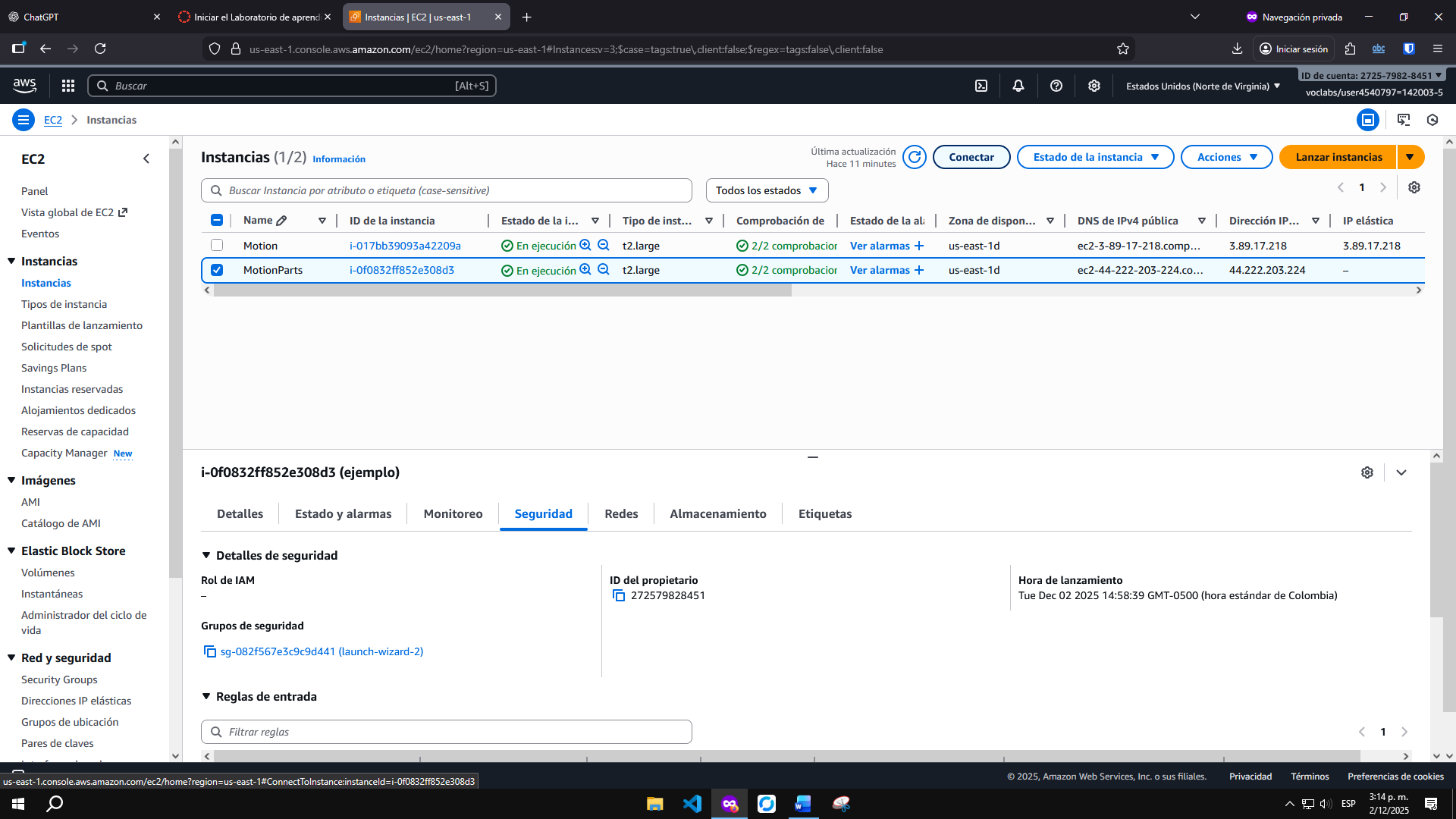
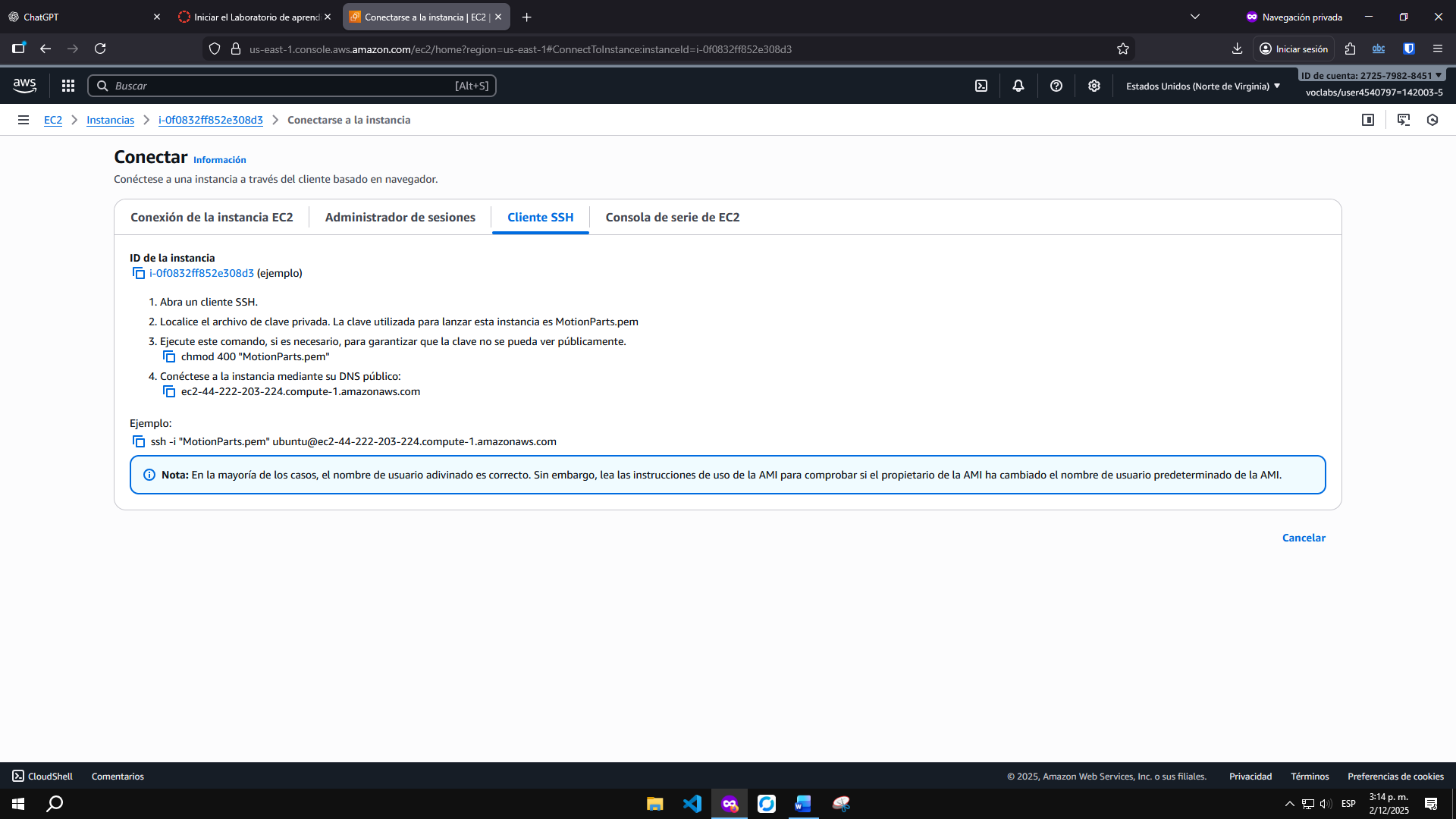
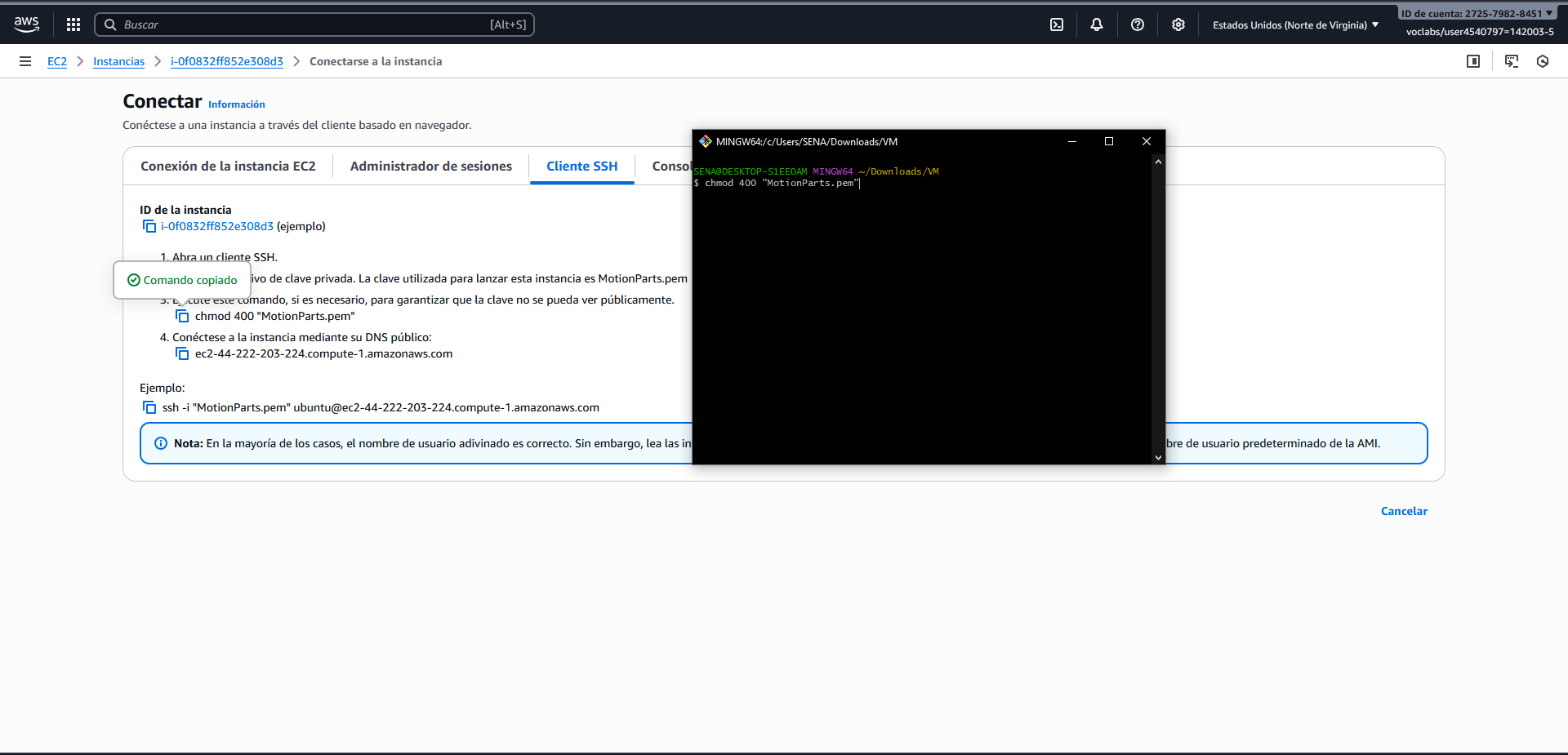
En la sección “configurar almacenamiento” se debe dejar como lo muestra la imagen a continuación:  
  
  
Antes de lanzar la instancia, verificamos el resumen para asegurarnos de que todo coincide perfectamente con lo hecho en los pasos anteriores:  
  
Si todo es correcto, se puede dar click al botón de “Lanzar instancia”  
  
  
  
  
  
  
  
  
  
  
  
  
Esperar a que se inicialice correctamente, debe aparecer como la imagen a continuación:   
  
  
La instancia se creará e inicializará “comprobación de estado”, esto tomará unos minutos  
  
  
Una vez “comprobación de estado” pase de “inicializando” a “2/2 comprobaciones superadas” la instancia estará lista para usarse:  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**3. Configuración de grupo de seguridad**1.Seleccione la instancia creada abajo aparecerán varias opciones y especificaciones de la instancia, seleccione Seguridad, la instancia tiene un grupo de seguridad asignado  
  
  
Le das click al grupo de seguridad o te diriges al panel ubicado en la izquierda, en la sección “Red y seguridad” le das click a la opción “Security Groups” o “Grupos de seguridad”

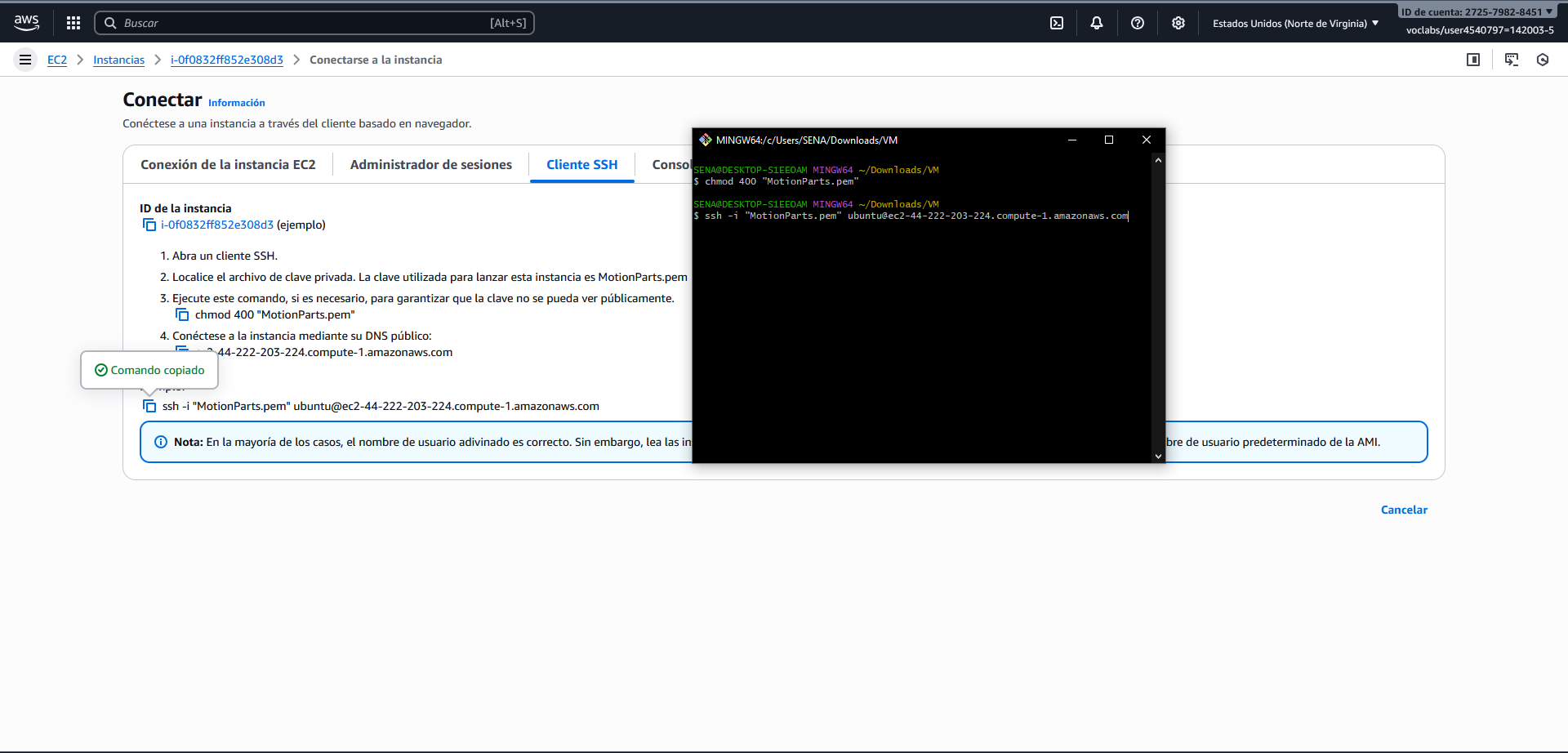
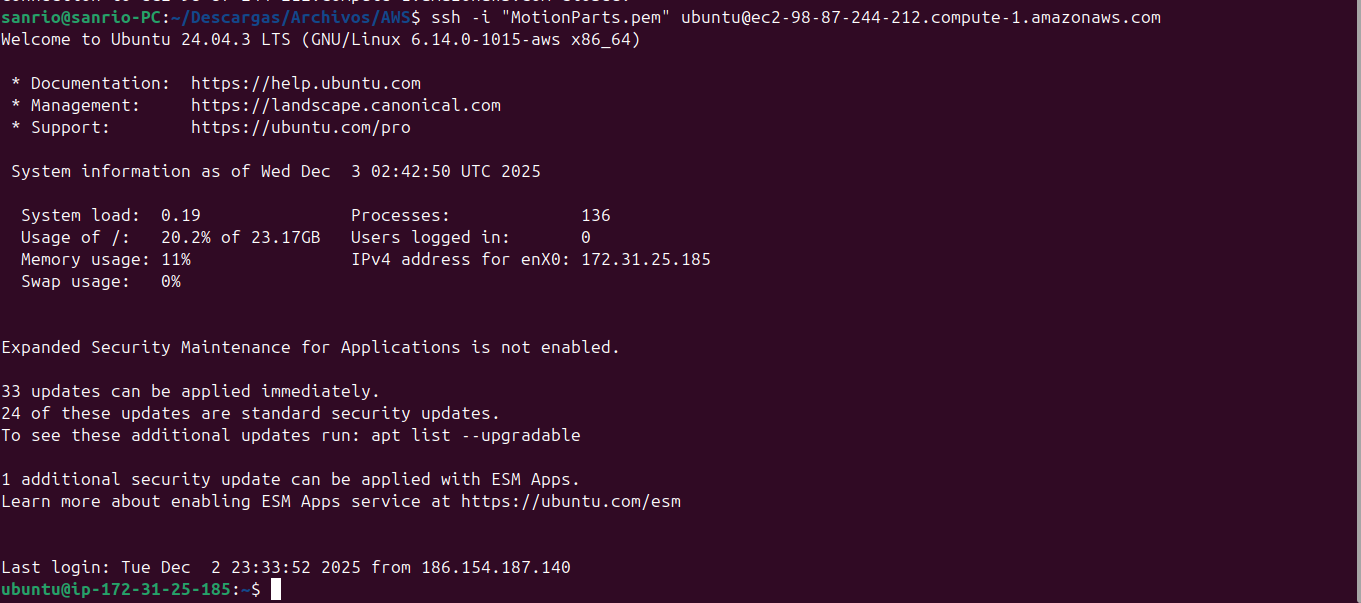
  
Seleccione el grupo de seguridad correspondiente y te vas a la opción de “Reglas de entrada” y editamos las reglas de entrada  
  
  
2. Al editar las reglas de entrada se deben agregar unas reglas específicas del proyecto para que funcione correctamente  
  
Le das al botón “Agregar regla” con la siguiente configuración  
Tipo: TCP personalizado  
Intervalo de puertos: puertos del proyecto (4200, 8080, 9000, 80, 22, 443)  
Origen: Anywhere IPv4  
Descripción: Mencionar a qué se dedica el puerto

  
  
Debe verse de la siguiente manera:  
  
Le das al botón “Guardar reglas” y el grupo de seguridad estaría configurado  
  
  
  
  
  
  
  
  
  
**4. Configuración de IP elastica**Configurar una IP elástica (fija) es importante porque AWS suele cambiar la IP a la cual puedes acceder, para facilitar esto se asigna una IP estática a continuación  
  
1. En la sección “Red y seguridad” te diriges al apartado “Direcciones IP elásticas” o “Elastic IP” y aparecerá la pantalla a continuación  
  
Le das al botón “Asignar dirección IP elástica”  
  
  
En la pantalla a continuación dar click al botón “Asignar” para crear la nueva dirección IP

  
  
  
  
  
  
  
  
  
  
  
2. Verificas que la dirección haya sido creada, si es asi seleccionas la dirección correspondiente y te vas al botón “Acciones” y se desplegará un menú, le das a la opción “Dirección IP elástica asociada”  
  
Te redirigirá a la siguiente pantalla, en la sección “Instancia” seleccione la instancia que creamos, en la sección “Dirección IP privada” se selecciona la única opción disponible y le das al botón “Asociado”

  
  
La asignación fue exitosa si aparece el mensaje a continuación

  
  
  
**5. Conectarse a la instancia**1. Una vez configurada la instancia, para conectarse se selecciona la instancia a la que se desea conectar y le das al botón “Conectar”  
  
  
Se redireccionará a esta pantalla, seleccione la opción “Cliente SSH”  
  
  
  
2. Se abre una terminal Git Bash en la ubicación donde se almacenó el archivo .pem  
Ejecuta este comando, si es necesario, para garantizar que la clave no se pueda ver públicamente.  
chmod 400 "MotionParts.pem"  
  
  
Conéctese a la instancia mediante su DNS público mediante el siguiente comando brindado por AWS: ssh -i "MotionParts.pem" ubuntu@ec2-98-87-244-212.compute-1.amazonaws.com

  
**Nota**: Para acceder desde windows, usar una terminal [Git Bash](https://git-scm.com/install/windows)  
  
La conexión fue exitosa si te aparece lo siguiente

# Instalación de Docker en Ubuntu

### **Paso 1: Actualizar el Sistema**

sudo apt-get update

### **Paso 2: Instalar Dependencias**

sudo apt-get install ca-certificates curl

### **Paso 3: Crear Directorio para Llaves**

sudo install -m 0755 -d /etc/apt/keyrings

### **Paso 4: Agregar la Llave GPG de Docker**

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

### **Paso 5: Agregar el Repositorio de Docker**

1.  
echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

2. sudo apt-get update

### **Paso 6: Instalar Docker**

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

### **Paso 7: Verificar la Instalación**

docker -v

Debería mostrar la versión de Docker instalada.  
Ejemplo:  
Docker version 29.1.2, build 890dcca

### **Paso 8: Configurar Permisos de Usuario (Opcional)**

Si al ejecutar docker ps aparece un error de permisos, tiene dos opciones:

**Opción A:** Usar sudo antes de cada comando docker

**Opción B:** Agregar su usuario al grupo docker (recomendado)

sudo usermod -aG docker ubuntu

Para aplicar los cambios sin reiniciar:

newgrp docker

O reinicie la sesión/máquina para que los cambios tengan efecto.

# **Configuración del Proyecto**

### **Crear el Archivo docker-compose.yml**

Cree una nueva carpeta llamada “motionparts”, usando “mkdir motionparts && cd motionparts”  
Cree un archivo llamado docker-compose.yml usando “touch docker-compose.yml && nano docker-compose.yml” dentro de la carpeta “motionparts” del proyecto con el siguiente contenido:

services:  
   
portainer-ce:

image: portainer/portainer-ce:latest

container\_name: portainer-ce

restart: unless-stopped

command: -H unix:///var/run/docker.sock

ports:

- "9000:9000"

volumes:

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

- portainer\_data:/data

db:

build: ./db

image: san2rio/motionparts:db-latest

container\_name: motionparts\_mongo

restart: always

environment:

MONGO\_INITDB\_DATABASE: motionparts

volumes:

- mongo\_data:/data/db

networks:

- motionparts\_net

healthcheck:

test: ["CMD", "mongosh", "--eval", "db.adminCommand('ping')"]

interval: 10s

timeout: 5s

retries: 5

start\_period: 40s

backend:

build: ./Backend

image: san2rio/motionparts:backend-latest

container\_name: motionparts\_backend

restart: always

depends\_on:

db:

condition: service\_healthy

environment:

SPRING\_DATA\_MONGODB\_URI: mongodb://db:27017/motionparts

ports:

- "8080:8080"

volumes:

- ./Backend/uploads:/app/uploads

networks:

- motionparts\_net

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:8080/api/dev/ping"]

interval: 10s

timeout: 5s

retries: 5

start\_period: 60s

frontend:

build: ./Frontend

image: san2rio/motionparts:frontend-latest

container\_name: motionparts\_frontend

restart: always

depends\_on:

backend:

condition: service\_healthy

ports:

- "4200:80"

networks:

- motionparts\_net

volumes:

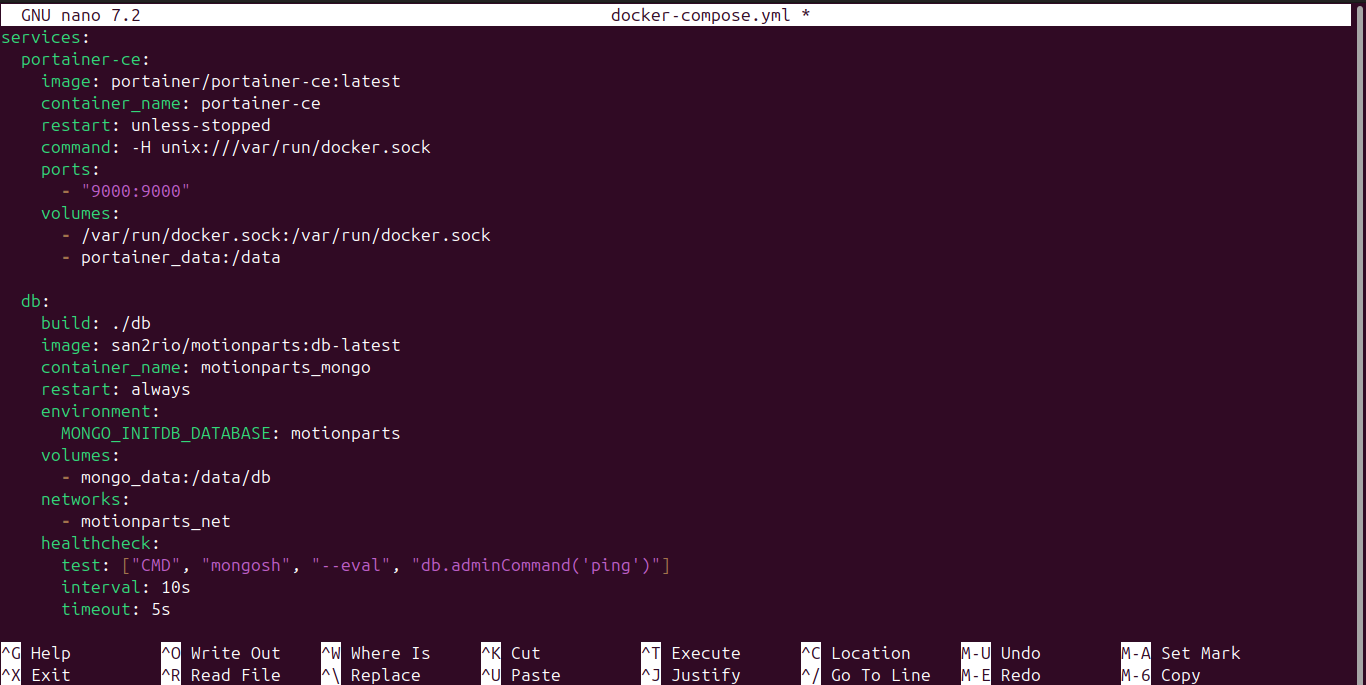
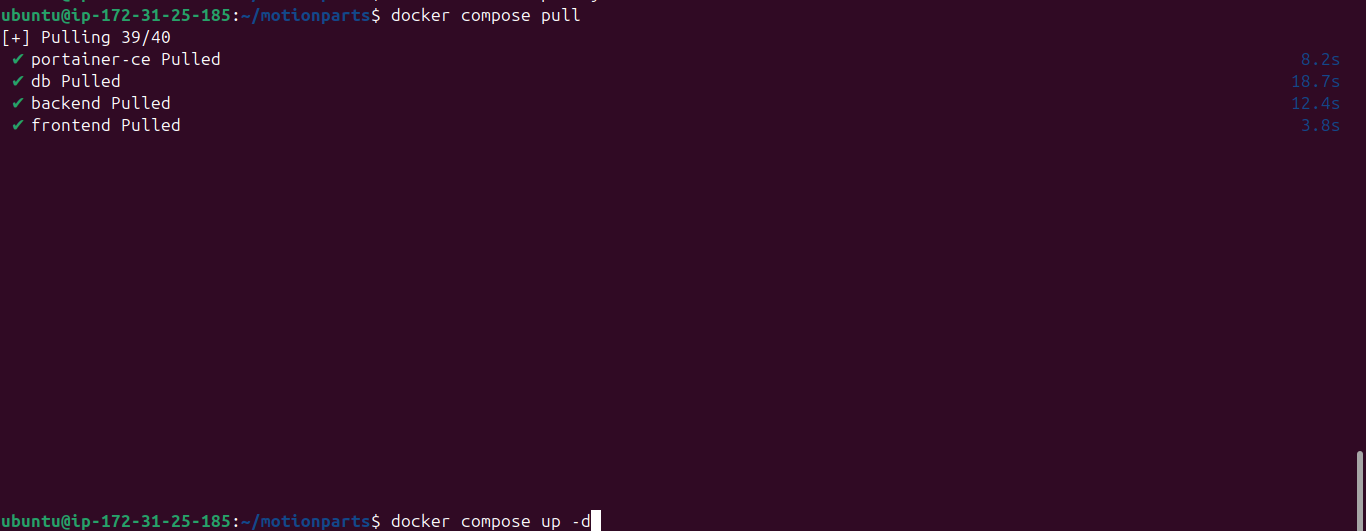
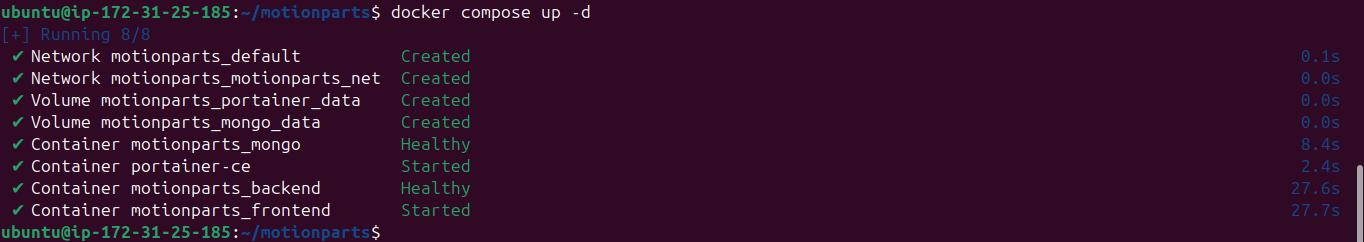
mongo\_data:

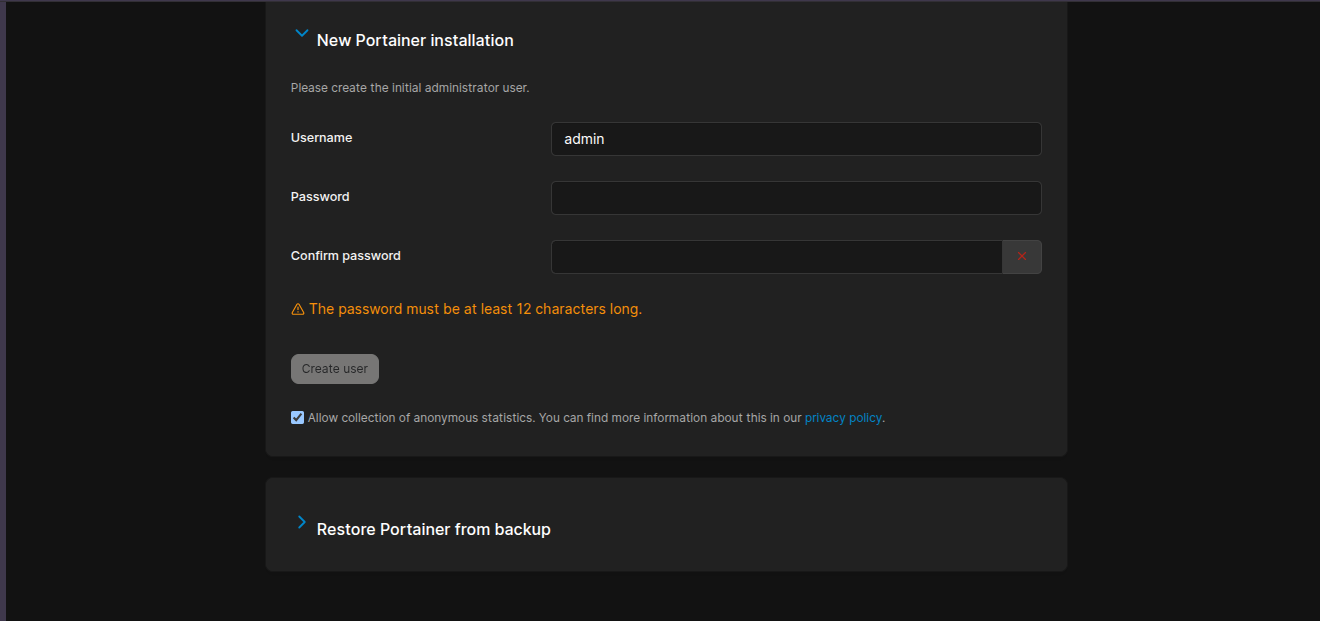
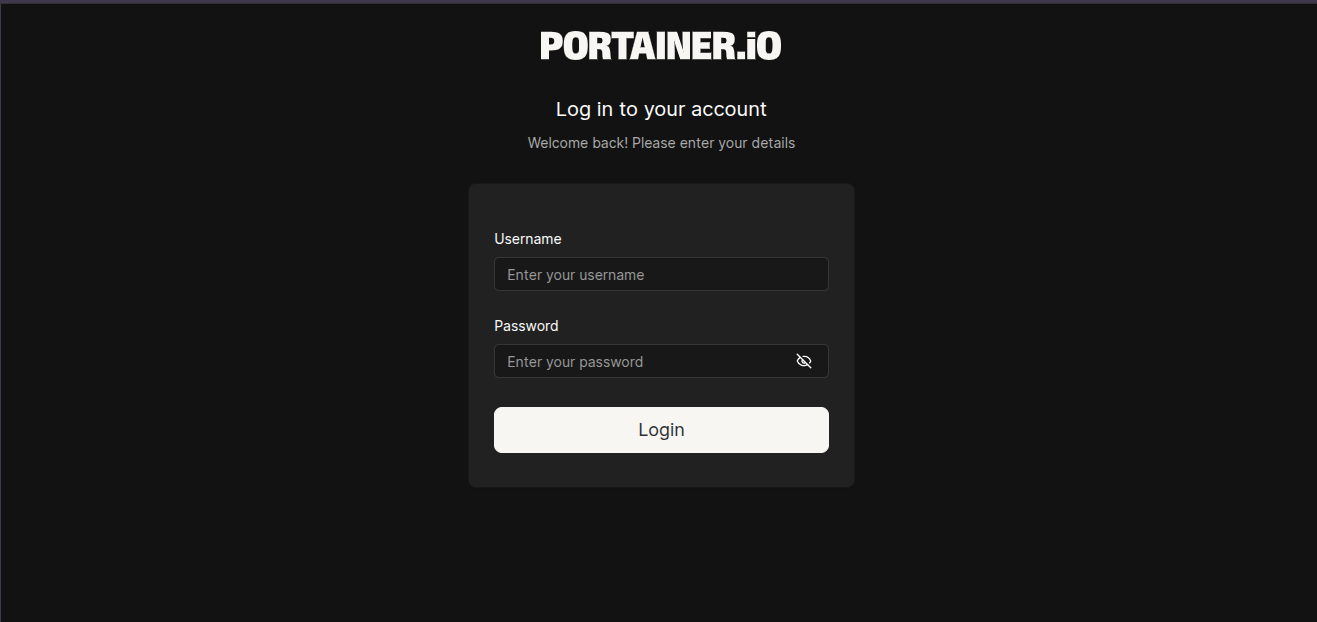
portainer\_data:

driver: local

networks:

motionparts\_net:

  
Se da Ctrl + O para guardar y Ctrl + X para salir del editor  
  
  
**Construir y Levantar los Contenedores**Para usar las imágenes desde docker hub solo se necesitaran dos comandos docker  
  
“docker compose pull” este comando trae las imágenes del repositorio san2rio/motionparts  
  
“docker compose up -d” esto levantara los 4 contenedores (Mongo, Backend, Frontend y Portainer)  
  
Para verificar el estado de los contenedores se ejecuta el comando “docker ps” o “docker ps -a”  
  
  
Otra opción es Portainer:  
Portainer es una herramienta de gestión visual para contenedores Docker.

1. Abra su navegador y acceda a: http://[IP-del-servidor]:9000
2. En el primer acceso, cree una cuenta de administrador  
   
3. Si ya existe una cuenta, solicite las credenciales al administrador del sistema  
   Desde Portainer puede monitorear el estado de los contenedores, ver logs y realizar tareas administrativas sin necesidad de acceder directamente al servidor

# **Certificado SSL Instalación de Caddy**

Caddy es un servidor web moderno que gestiona automáticamente los certificados SSL/TLS mediante Let's Encrypt.  
  
**Nota**: para sacar este certificado es necesario tener un dominio asignado al proyecto, el nombre de estos puede variar según las páginas y herramientas que se usen, hay páginas que tienen un proceso propio de certificación SSL

### **Paso 1: Instalar Dependencias**

sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https curl

### **Paso 2: Agregar la Llave GPG de Caddy**

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o /usr/share/keyrings/caddy-stable-archive-keyring.gpg

### **Paso 3: Agregar el Repositorio de Caddy**

curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-stable.list

### **Paso 4: Configurar Permisos**

chmod o+r /usr/share/keyrings/caddy-stable-archive-keyring.gpg

chmod o+r /etc/apt/sources.list.d/caddy-stable.list

### **Paso 5: Instalar Caddy**

sudo apt update

sudo apt install caddy

### **Paso 6: Configurar el Caddyfile**

Navegue al directorio de configuración de Caddy:

cd /etc/caddy

Verifique que existe el archivo Caddyfile: con el comando ls  


Edite el archivo Caddyfile: sudo nano Caddyfile

Reemplace el contenido con la siguiente configuración (sustituya dominio.org por su dominio real):

dominio.org {

encode gzip

log

# Proxy API requests to backend on host port 8080

reverse\_proxy /api/\* http://127.0.0.1:8080

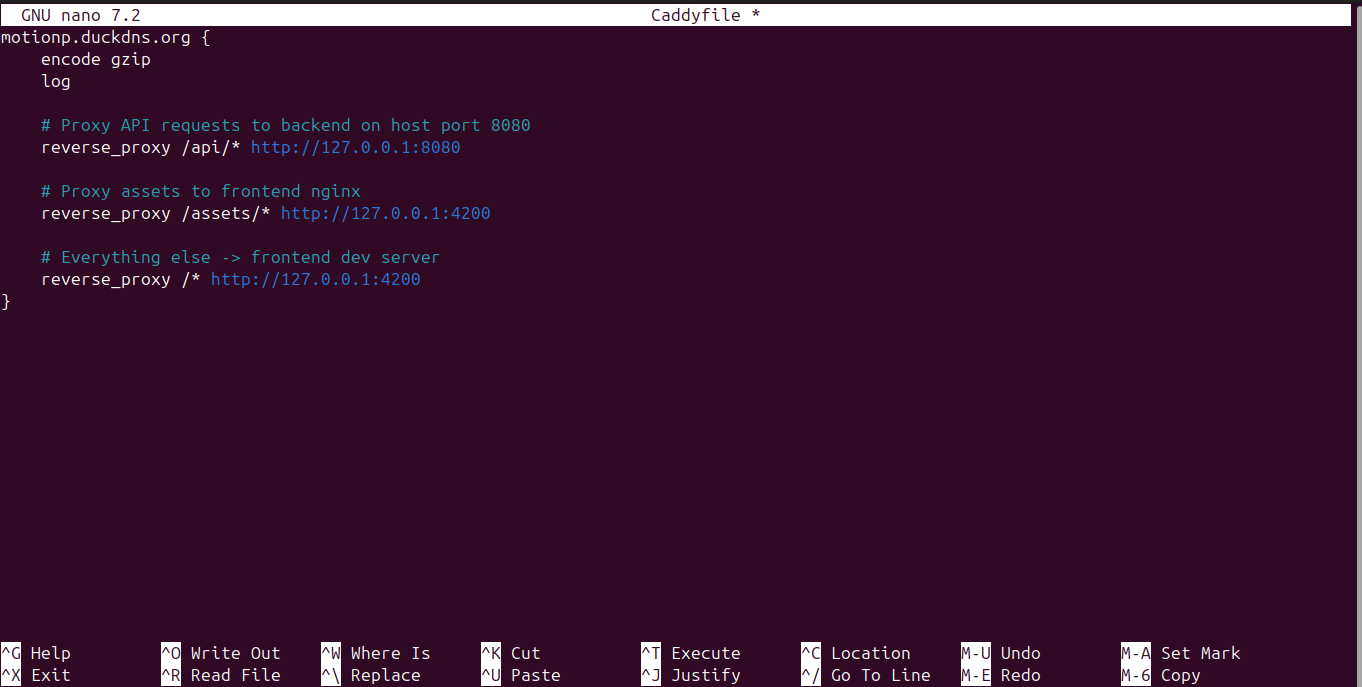
# Proxy assets to frontend nginx

reverse\_proxy /assets/\* http://127.0.0.1:4200

# Everything else -> frontend dev server

reverse\_proxy /\* http://127.0.0.1:4200

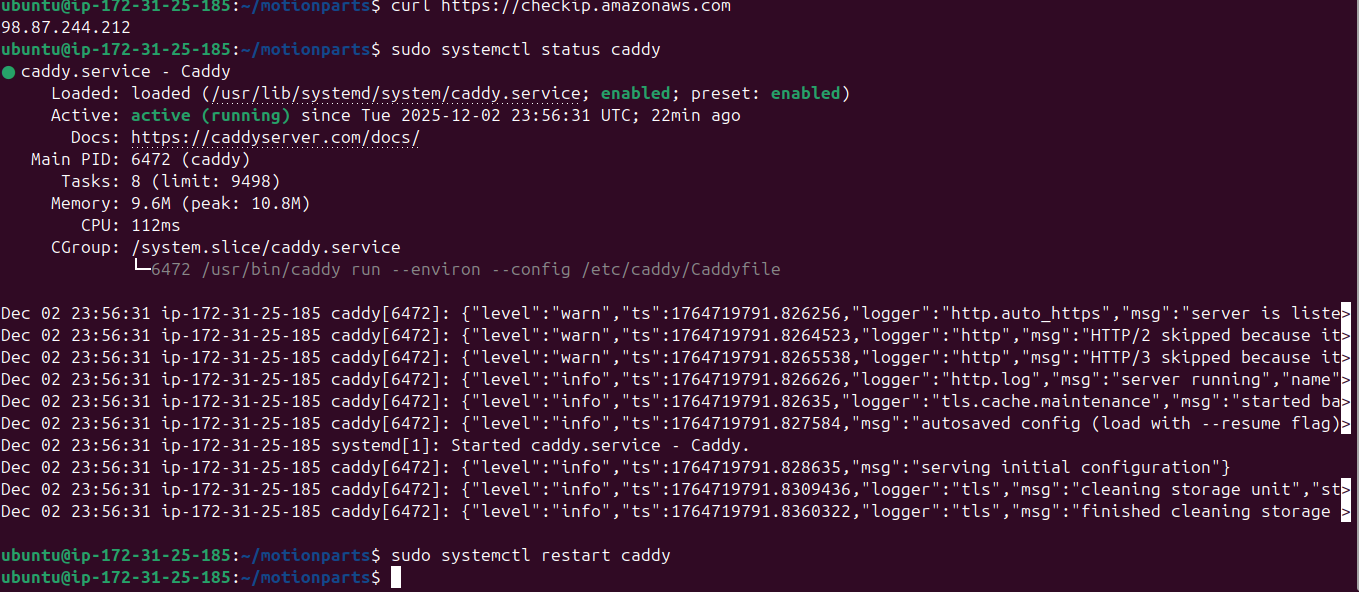
}

Guarde el archivo (Ctrl + O, Enter, Ctrl + X en nano).  
  


### **Paso 7: Reiniciar Caddy**

sudo systemctl restart caddy

### **Paso 8: Verificar el Estado de Caddy**

sudo systemctl status caddy  


**Importante:** Caddy tardará de 3 a 5 minutos en generar y configurar automáticamente el certificado SSL de Let's Encrypt. Durante este tiempo, el sitio puede ser inaccesible temporalmente.

# **Acceso y Verificación**

Una vez completada la instalación, el sistema estará disponible en:

### **URLs de Acceso**

* **Aplicación Principal:** https://dominio.org
* **Portainer:** http://[IP-del-servidor]:9000

### **Verificación del Certificado SSL**

Después de aproximadamente 5 minutos, verifique que el sitio esté accesible mediante HTTPS con el candado verde en el navegador, indicando que el certificado SSL es válido.

El cambio será de http://dominio.org → https://dominio.org   
  
**Comandos Útiles**

**Ver logs de los contenedores:**

docker compose logs -f [nombre\_servicio]

**Reiniciar un servicio específico:**

docker compose restart [nombre\_servicio]

**Detener todos los servicios:**

docker compose down

**Actualizar y reiniciar servicios:**

docker compose up -d --build

**Ver logs de Caddy:**

sudo journalctl -u caddy -f