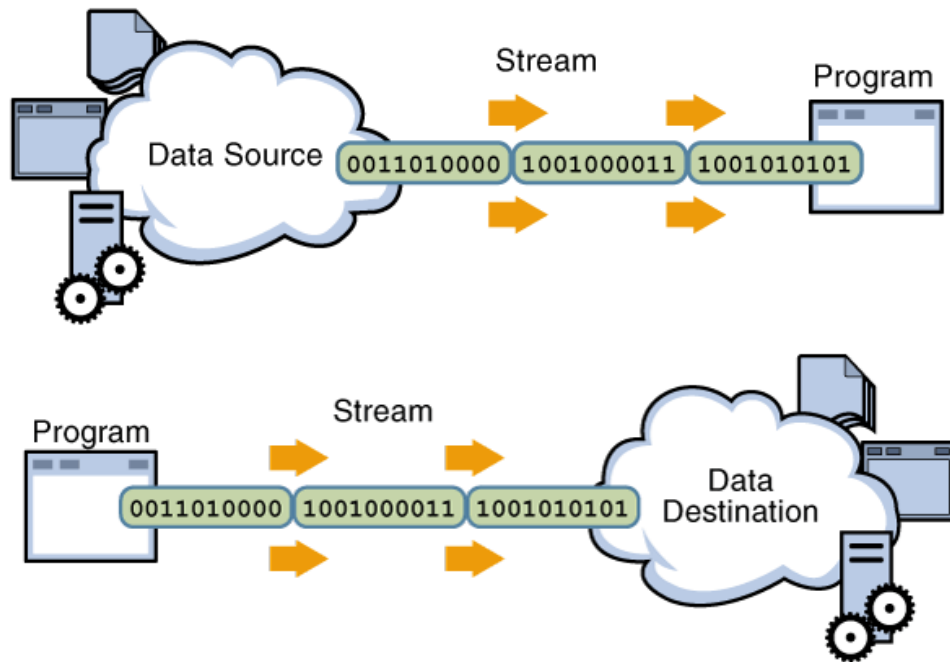


# WebSockets de Java en la nube (AWS)



## 1. Introducción

## 2. Requisitos

## 3. Guía paso a paso

### 3. 1. Preparar el entorno de la nube

3. 1. 1. Iniciar Laboratorio

3. 1. 2. Crear un entorno Cloud9

3. 1. 3. Creación del servidor de Sockets

3. 1. 4. Abrir el puerto en la instancia EC2 del cloud9

3. 1. 5. Dirección pública de la EC2

### 3. 2. Preparar el cliente local

### 3. 3. Ejecución de prueba

3. 3. 1. Desde el punto de vista del cliente

3. 3. 2. Desde el punto de vista del servidor

## 4. Fuentes de información

# 1. Introducción

---

La intención de este documento es la de dar una perspectiva más realista del uso de sockets, ya que en lugar de usar la misma máquina del alumno como cliente y servidor, vamos a desplegar el servidor del socket en una máquina alojada en la nube de Amazon (AWS).

## 2. Requisitos

---

Para realizar esta práctica guiada necesitamos:

- Acceso al Learner Lab proporcionado por el profesor. (<https://awsacademy.instructure.com>)
- Conocimientos sobre los websockets, IP's y puertos.
- Un dispositivo local con capacidad de ejecutar un cliente de socket, con acceso a los puertos e Ip's de AWS (Ojo con la red de conselleria)

## 3. Guía paso a paso

### 3.1. Preparar el entorno de la nube

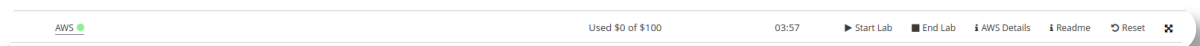
#### 3.1.1. Iniciar Laboratorio

Lo primero que necesitamos es arrancar el laboratorio, para ello Accedemos al LMS del awsacademy, buscamos el Curso facilitado por el docente, accedemos a sus contenidos y a continuación al Learner Lab. (Si es la primera vez que accedemos debemos aceptar los términos de uso).

Inicialmente el laboratorio está en rojo:

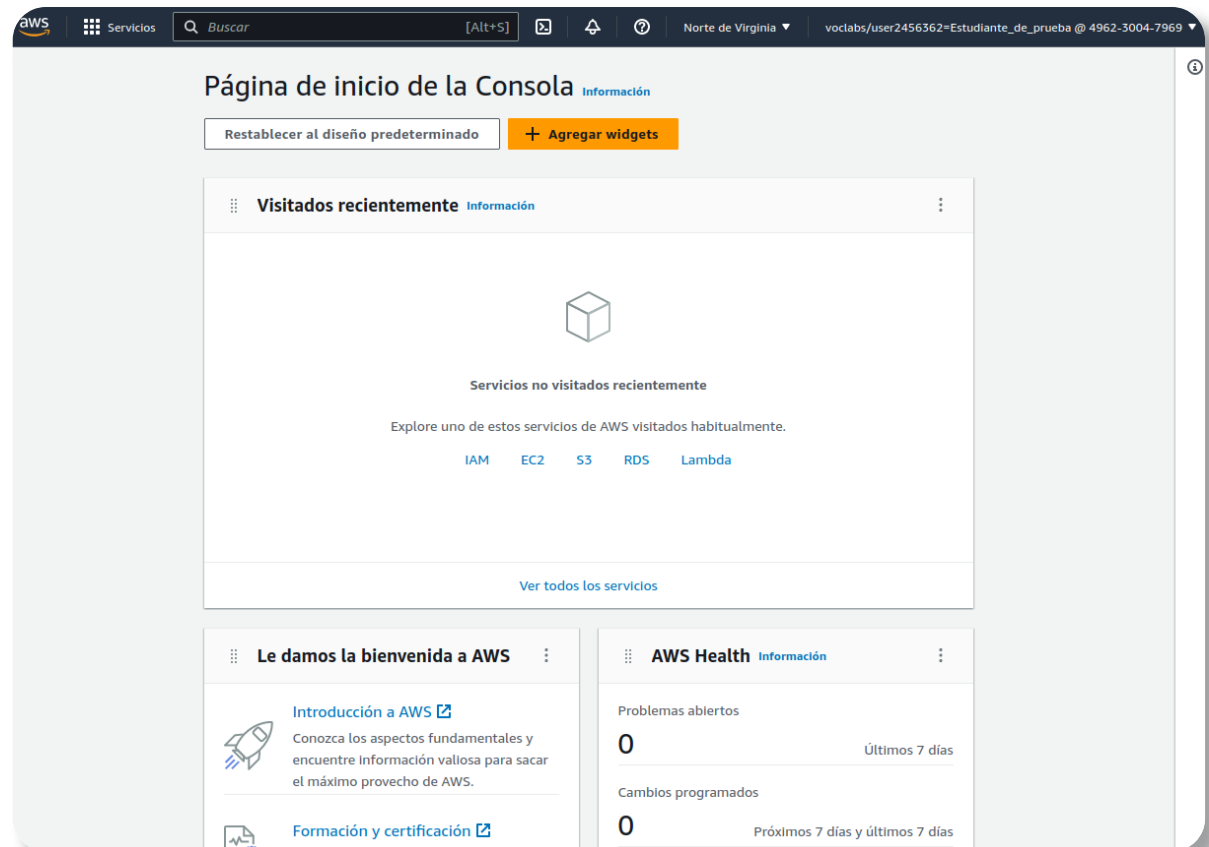


Elegimos la opción `Start Lab` y esperamos a que aparezca el laboratorio en verde:



Por defecto el Learner Lab nos proporciona 100 dolares de saldo, y un tiempo de 4 horas, tras el cual se detendrán la mayoría de servicios que tengamos en marcha. Pero mientras quede saldo podemos volver a iniciar el Laboratorio y dispondremos de 4 horas más.

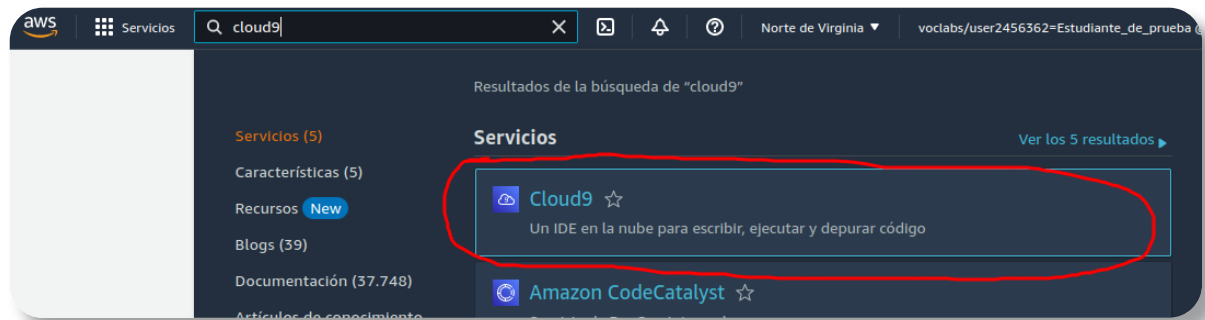
Una vez aparece en verde podemos hacer click sobre las letras AWS y aparecerá el Dashboard de AWS (debemos permitir las ventanas emergentes):



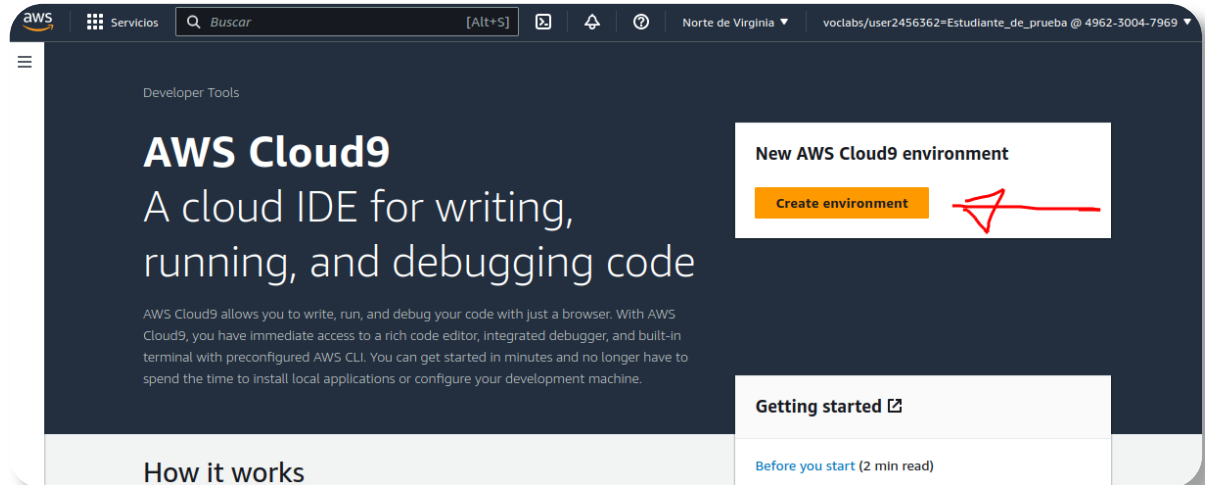
#### 3.1.2. Crear un entorno Cloud9

Cloud9 es un entorno de desarrollo en la nube que proporciona AWS asociado a una instancia EC2 (máquina virtual en la nube).

El primer paso es crear este entorno, para ello buscamos Cloud9 en la parte superior del Dashboard:



A continuación seleccionamos `Create environment`:



En la siguiente ventana debemos especificar el **nombre** (`Name`), cambiaremos la **plataforma** a `Ubuntu Server 22.04 LTS`, también podemos ampliar el tiempo de Timeout para no tener problemas a `4 horas` i por último dentro de los **Network settings** elegiremos la conexión por `SSH`, el resto de opciones se quedan por defecto y pulsamos el botón naranja del final `Create`.

Crear entorno [Info](#)

## Detalles

## Nombre

cloud9David

Límite de 60 caracteres alfanuméricos y únicos por usuario.

Descripción: *opcional*

Límite de 200 caracteres.

Tipo de entorno [Info](#)

Determina en qué se ejecutará el IDE de Cloud9.

☒ Nueva instancia de EC2

Cloud9 crea una instancia de EC2 en su cuenta. Cloud9 no puede cambiar la configuración de la instancia de EC2 después de crearla.

☐ Computación existente

Ya tiene una instancia o un servidor que desea usar.

## Nueva instancia de EC2

Tipo de instancia [Info](#)

La memoria y la CPU de la instancia de EC2 que se creará para que se ejecute Cloud9.

☒ t2.micro (1 GIB RAM + 1 vCPU)

Apto para el nivel gratuito. Ideal para usuarios educativos y de exploración.

☐ t3.small (2 GIB RAM + 2 vCPU)

Recomendado para proyectos web pequeños.

☐ m5.large (8 GIB RAM + 2 vCPU)

Recomendado para la producción y el desarrollo de uso más general.

☐ Tipos de instancias adicionales

Explore instancias adicionales que se ajusten a sus necesidades.

Plataforma [Info](#)

Se instalará en su instancia de EC2. Recomendamos Amazon Linux 2023.

Ubuntu Server 22.04 LTS

## Tiempo de espera

Cuánto tiempo puede permanecer inactivo Cloud9 (sin intervención del usuario) antes de hibernar automáticamente. Esto ayuda a evitar cargos innecesarios.

4 horas

Configuración de red [Info](#)

## Conexión

Cómo se accede a su entorno.

☐ AWS Systems Manager (SSM)

Accede al entorno a través de SSM sin abrir los puertos entrantes (sin entrada).

☒ Secure Shell (SSH)

Accede al entorno directamente a través de SSH, abre los puertos entrantes.

► Configuración de VPC [Info](#)► Etiquetas: *opcional* [Info](#)

Las etiquetas se asignan a los recursos de AWS. Cada etiqueta consta de una clave y un valor opcional. Puede utilizar etiquetas para buscar y filtrar sus recursos o realizar un seguimiento de sus costos de AWS.



## Se crearán los siguientes recursos de IAM en su cuenta

- **AWSServiceRoleForAWSCloud9**- AWS Cloud9 crea un rol vinculado a un servicio para usted. Esto permite que AWS Cloud9 invoque a otros servicios de AWS en su nombre. Puede eliminar el rol desde la consola de AWS IAM cuando ya no tenga ningún entorno de AWS Cloud9. [Más información](#)

Cancelar

Crear

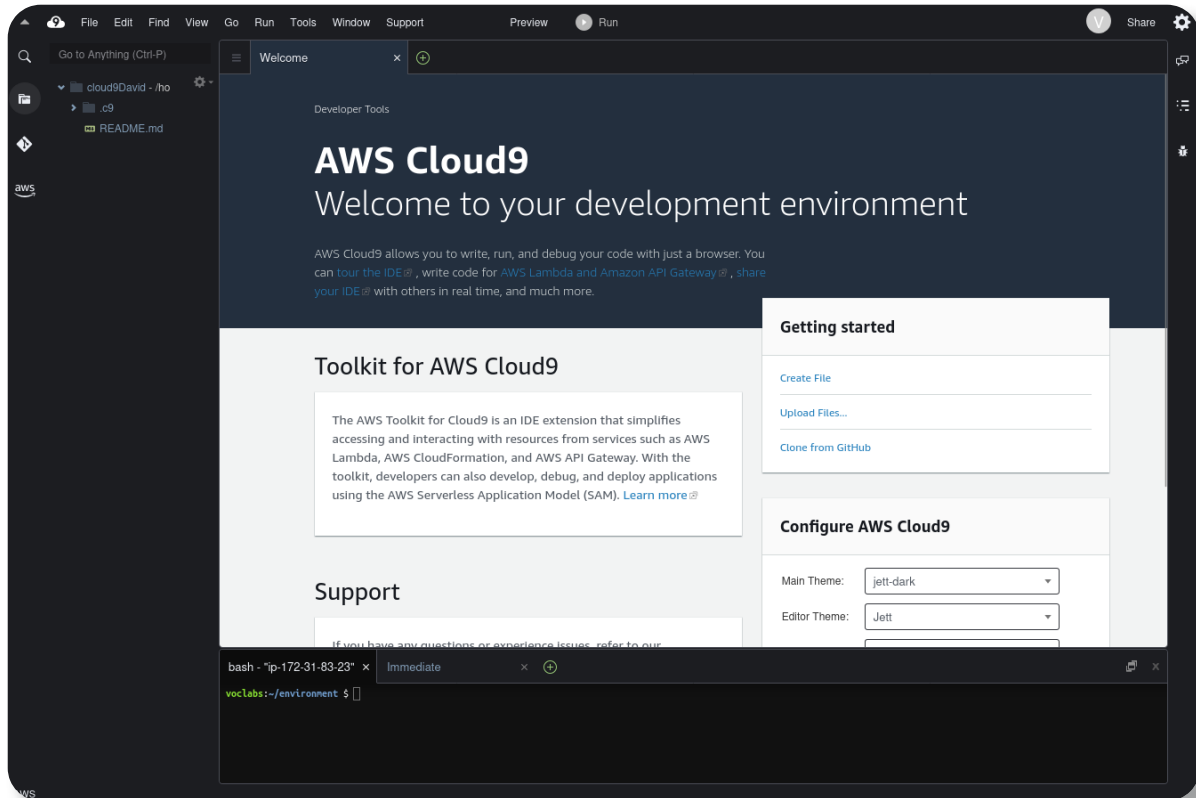
Si todo ha ido bien podemos seleccionar el botón **Open** :

Environments (1) Delete View details Open in Cloud9 Create environment

My environments

	Name	Cloud9 IDE	Environment type	Connection	Permission	Owner ARN
<input type="radio"/>	cloud9David	Open	EC2 Instance	Secure Shell (SSH)	Owner	arn:aws:sts::496230047969:assumed-role/voclabs/user2456362=Estudiante_de_prueba

Y deberíamos ver algo parecido a esto:



### 3.1.3. Creación del servidor de Sockets

Primero cerraremos la ventana de bienvenida, a continuación creamos un nuevo fichero, por ejemplo `AWSServerSocket.java` con el siguiente código java:

```

1  import java.io.*;
2  import java.net.*;
3
4  public class AWSServerSocket {
5
6      private static final int PORT=11000;
7
8      public static void main(String[] args) throws IOException, ClassNotFoundException {
9          String FraseClient;
10         String FraseMajuscules;
11         ServerSocket serverSocket;
12         Socket clientSocket;
13         ObjectInputStream entrada;
14         ObjectOutputStream eixida;
15         serverSocket = new ServerSocket(PORT);
16         System.out.println("Server iniciado y escuchando en el puerto "+ PORT);
17         while (true) {
18             clientSocket = serverSocket.accept();
19             entrada = new ObjectInputStream(clientSocket.getInputStream());
20             FraseClient = (String) entrada.readObject();
21
22             System.out.println("La frase recibida es: " + FraseClient);
23
24             eixida = new ObjectOutputStream(clientSocket.getOutputStream());
25             FraseMajuscules = FraseClient.toUpperCase();
26             System.out.println("El server devuelve la frase: " + FraseMajuscules);
27             eixida.writeObject(FraseMajuscules);
28
29             clientSocket.close();
30             System.out.println("Server esperando una nueva conexión...");
31         }
32     }
33 }
```

Debería quedar algo así:



```

1  import java.io.*;
2  import java.net.*;
3
4  public class ServidorSocket {
5
6      private static final int PORT=11000;
7
8      public static void main(String[] args) throws IOException, ClassNotFoundException {
9          String FraseClient;
10         String FraseMajuscules;
11         ServerSocket serverSocket;
12         Socket clientSocket;
13         ObjectInputStream entrada;
14         ObjectOutputStream eixida;
15         serverSocket = new ServerSocket(PORT);
16         System.out.println("Server iniciado y escuchando en el puerto " + PORT);
17         while (true) {
18             clientSocket = serverSocket.accept();
19             entrada = new ObjectInputStream(clientSocket.getInputStream());
20             FraseClient = (String) entrada.readObject();
21
22             System.out.println("La frase recibida es: " + FraseClient);
23
24             eixida = new ObjectOutputStream(clientSocket.getOutputStream());
25             FraseMajuscules = FraseClient.toUpperCase();
26             System.out.println("El server devuelve la frase: " + FraseMajuscules);
27             eixida.writeObject(FraseMajuscules);
28
29             clientSocket.close();
30             System.out.println("Server esperando una nueva conexión...");
31         }
32     }
33 }

```

Y si iniciamos el servidor:

bash - "ip-172-31-83-23" x Immediate x ServidorSocket.java - Run

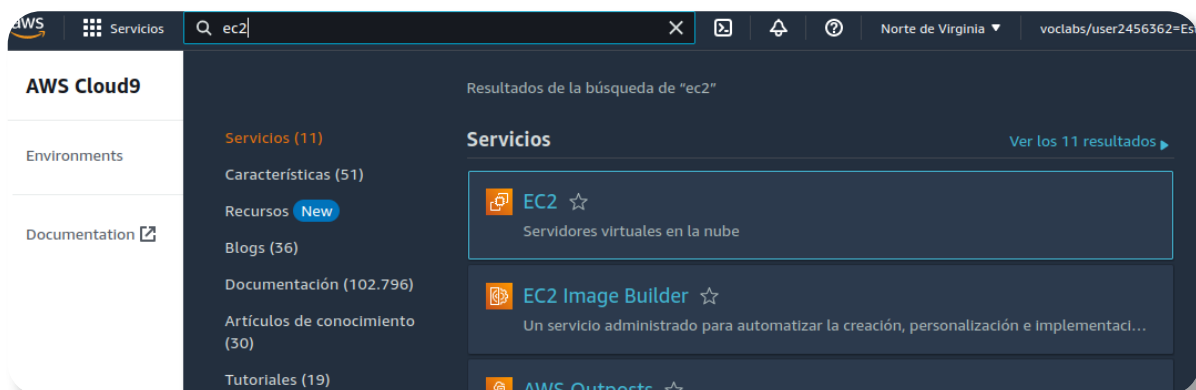
Stop Command: ServidorSocket.java

**Building ServidorSocket.java and running ServidorSocket**

Server iniciado y escuchando en el puerto 11000

### 3.1.4. Abrir el puerto en la instancia EC2 del cloud9

Ahora debemos volver a la pestaña donde tenemos el Dashboard de AWS y buscar EC2 (donde antes buscamos cloud9):



Una vez abierto elegimos la opción `Instancias (en ejecución)`:

**Recursos**

Actualmente, utiliza los siguientes recursos de Amazon EC2 en la región EE.UU. (Virginia):

Instancias (en ejecución)	1	Auto Scaling Groups
Balanceadores de carga	0	Direcciones IP elásticas
Grupos de seguridad	2	Grupos de ubicación

Deberíamos tener al menos una Instancia, si tenemos más de una debemos buscar la que contenga el nombre de nuestra instancia cloud9, debemos marcar el check que tiene justo delante del nombre y a continuación elegir la pestaña **Seguridad**:

**Instancias (1/1) Información**

Buscar instancia por atributo o etiqueta (case-sensitive)

Estado de la instancia = running X Quitar los filtros

<input checked="" type="checkbox"/>	Name	ID de la instancia	Estado de la i...	Tipo de inst...	Comprobación ...
<input checked="" type="checkbox"/>	aws-cloud9-cloud9David-83bd4fb6e5bf401e9...	i-069949e33eb20baa8	En ejecución	t2.micro	2/2 comprobación

**Instancia: i-069949e33eb20baa8 (aws-cloud9-cloud9David-83bd4fb6e5bf401e9b81d1e6f35291c5)**

Detalles Seguridad Redes Almacenamiento Comprobaciones de estado Monitoreo Etiquetas

**Detalles de seguridad**

Rol de IAM: - ID del propietario: 496230047969

Grupos de seguridad: sg-0ab93f95e05631a5c (aws-cloud9-cloud9David-83bd4fb6e5bf401e9b81d1e6f35291c5-InstanceSecurityGroup-7JT461Z8I2RR)

**Reglas de entrada**

Si nos fijamos en las reglas de entrada del grupo de seguridad, solo tiene habilitada la entrada para el puerto 22 (SSH), a continuación hacemos click sobre el enlace del Grupo de seguridad:

#### Grupos de seguridad

[sg-0ab93f95e05631a5c \(aws-cloud9-cloud9David-83bd4fb6e5bf401e9b81d1e6f35291c5-InstanceSecurityGroup-7JT461Z8I2RR\)](#)

#### Reglas de entrada

Q Filtrar reglas

Nombre	ID de la regla del grupo d...	Intervalo de pu...	Protocolo
-	sgr-0cb10956d0b223f9d	22	TCP
-	sgr-09bd2851a6631f7da	22	TCP

Y añadiremos el puerto 11000 (o el que hayamos elegido para nuestro servidor) a las reglas de entrada, elegimos el botón **Editar reglas de entrada**, a continuación **Agregar regla**. Elegimos **TCP Personalizado**, puerto 11000 y **AnywhereIpv4** y añadimos una descripción si lo deseamos:

## Editar reglas de entrada

Información

Las reglas de entrada controlan el tráfico entrante que puede llegar a la instancia.

ID de la regla del grupo de seguridad	Tipo	Protocolo	Intervalo de puertos	Origen	Descripción: opcional	
sgr-0cb10956d0b223f9d	SSH	TCP	22	Personalizada		Eliminar
sgr-09bd2851a6631f7da	SSH	TCP	22	Personalizada		Eliminar
-	TCP personalizado	TCP	11000	Anywhere-I...	SocketServer	Eliminar

Agregar regla

Cancelar Previsualizar los cambios Guardar reglas

Una vez hecho esto si volvemos a la pestaña Seguridad de nuestra instancia EC2 veremos la regla añadida.

### 3.1.5. Dirección pública de la EC2

Necesitamos saber la DNS de IPv4 pública de nuestra instancia EC2 para acceder desde el cliente, marcamos el check de nuestra instancia y accedemos a la primera pestaña **Detalles**, y nos fijamos en la parte derecha y pulsaremos el botón de copiar y guardaremos esta información para más adelante:

Instancias (1/1) Información

Buscar instancia por atributo o etiqueta (case-sensitive)

✓	Name	ID de la instancia	Estado de la i...	Tipo de inst...	Comprobación...	Estado de la ...	Zona de dispon...	DNS de IPv4 pública	Dirección IP...
✓	aws-cloud9-cloud9David-83bd4fb6e5bf401e9...	i-069949e33eb20baa8	En ejecución	t2.micro	2/2 comprobador	Sin alarmas	us-east-1c	ec2-3-84-52-97.comput...	3.84.52.97

Instancia: i-069949e33eb20baa8 (aws-cloud9-cloud9David-83bd4fb6e5bf401e9b81d1e6f35291c5)

Detalles Seguridad Redes Almacenamiento Comprobaciones de estado Monitoreo Etiquetas

▼ Resumen de instancia Información

ID de la instancia i-069949e33eb20baa8 (aws-cloud9-cloud9David-83bd4fb6e5bf401e9b81d1e6f35291c5)	Dirección IPv4 pública 3.84.52.97   dirección abierta	Direcciones IPv4 privadas 172.31.83.25
Dirección IPv6 -	Estado de la instancia En ejecución	DNS de IPv4 pública ec2-3-84-52-97.compute-1.amazonaws.com   dirección abierta
Tipo de nombre de anfitrión Nombre de IP: ip-172-31-83-23.ec2.internal	Nombre DNS de IP privada (solo IPv4) ip-172-31-83-23.ec2.internal	Direcciones IP elásticas -
Responder al nombre DNS de recurso privado -	Tipo de instancia t2.micro	Hallazgo de AWS Compute Optimizer Suscribirse a AWS Compute Optimizer para recibir recomendaciones.
Dirección IP asignada automáticamente 3.84.52.97 [IP pública]	ID de VPC vpc-0ba927e3f4ec4d112	

### 3.2. Preparar el cliente local

En nuestro IDE preferido creamos un nuevo archivo `ClienteSocket.java` con el siguiente código:

```
1 import java.io.*;
2 import java.net.Socket;
3 import java.util.Scanner;
4
5 public class AWSClienteSocket {
6
7     private final static int PUERTO = 11000;
8     private static final String DNSAWS = "ec2-54-173-21-231.compute-1.amazonaws.com";
9     //private static final String DNSAWS = "127.0.0.1";
10
11     public static void main(String[] args) throws IOException, ClassNotFoundException {
12         Scanner in = new Scanner(System.in);
13         System.out.print("Introduce la frase a enviar en minúsculas: ");
14         String frase = in.nextLine();
15
16         try (Socket socket = new Socket(DNSAWS, PUERTO)) {
17             ObjectOutputStream salida = new ObjectOutputStream(new
18                 BufferedOutputStream(socket.getOutputStream()));
19             System.out.println("Se envia la frase: " + frase);
20             salida.writeObject(frase);
21             salida.flush(); //vaciamos el buffer
22
23             ObjectInputStream entrada = new ObjectInputStream(new
24                 BufferedInputStream(socket.getInputStream()));
25             System.out.println("La frase recibida es: " + (String) entrada.readObject());
26         } catch (IOException ex) {
27             System.err.println("Error. De entrada salida.");
28         }
29     }
30 }
```

```

26 |     }
27 | }
28 |

```

Recuerda cambiar la constante `DNSAWS` por el `String` que corresponde con la dirección DNS IPv4 de tu instancia EC2 obtenida en el punto 3.1.5.

### 3.3. Ejecución de prueba

#### 3.3.1. Desde el punto de vista del cliente

Una vez ejecutado el cliente debe aparecer algo similar a esto:

```

1 | Introduce la frase a enviar en minúsculas

```

Escribimos nuestra frase, y al pulsar INTRO obtenemos el siguiente resultado:

```

1 | Introduce la frase a enviar en minúsculas
2 | esta frase está en minúsculas
3 | Se envía la frase esta frase está en minúsculas
4 | La frase recibida es: ESTA FRASE ESTÁ EN MINÚSCULAS

```

#### 3.3.2. Desde el punto de vista del servidor

La consola de salida del servidor por su parte debe haber registrado la conexión del cliente, la recepción de la frase, y la frase devuelta:

```

1 | Server iniciado y escuchando en el puerto 11000
2 | La frase recibida es: esta frase está en minúsculas
3 | El server devuelve la frase: ESTA FRASE ESTÁ EN MINÚSCULAS
4 | Server esperando una nueva conexión...

```

## 4. Fuentes de información

---

- <https://awsacademyinstructure.com>