

# Proyecto Parcial

## Cloud Computing

Integrantes:

- Chinchá León, Marcelo Andres - 202210092
- Céspedes Zevallos, Adrian Joshep - 202210088

UTEC  
Universidad  
de Ingeniería  
y Tecnología

# Índice

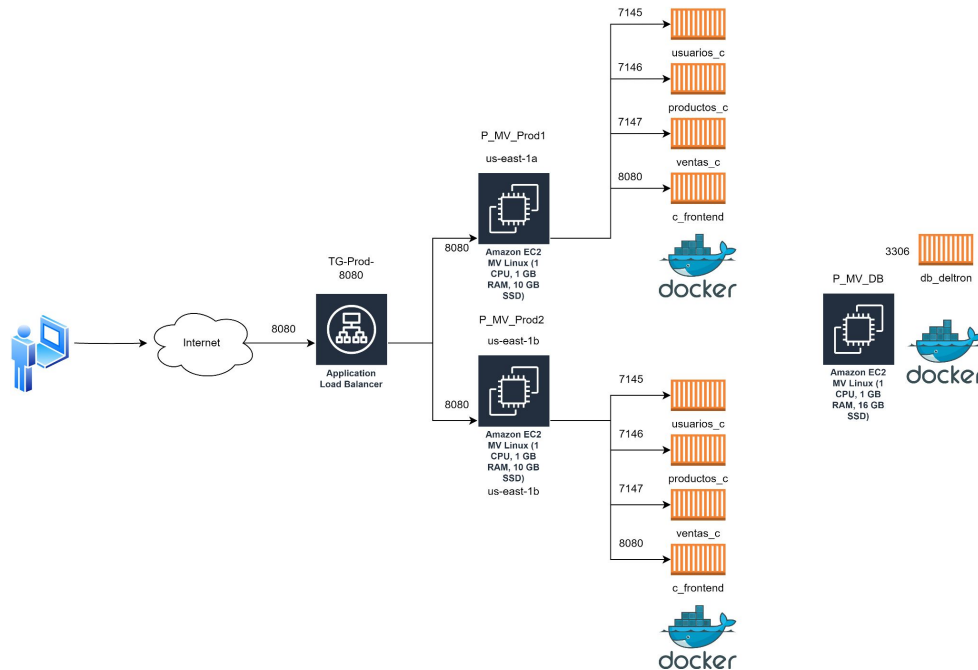
1. Estructura de proyecto
  - a. Diagrama de solución
  - b. Diseño de base de datos
  - c. Funcionalidades de microservicios
  - d. Front End
2. Despliegue
  - a. Requisitos para el despliegue
  - b. Pasos a seguir
  - c. Observaciones

# 1

## Estructura de proyecto

# Diagrama de solución

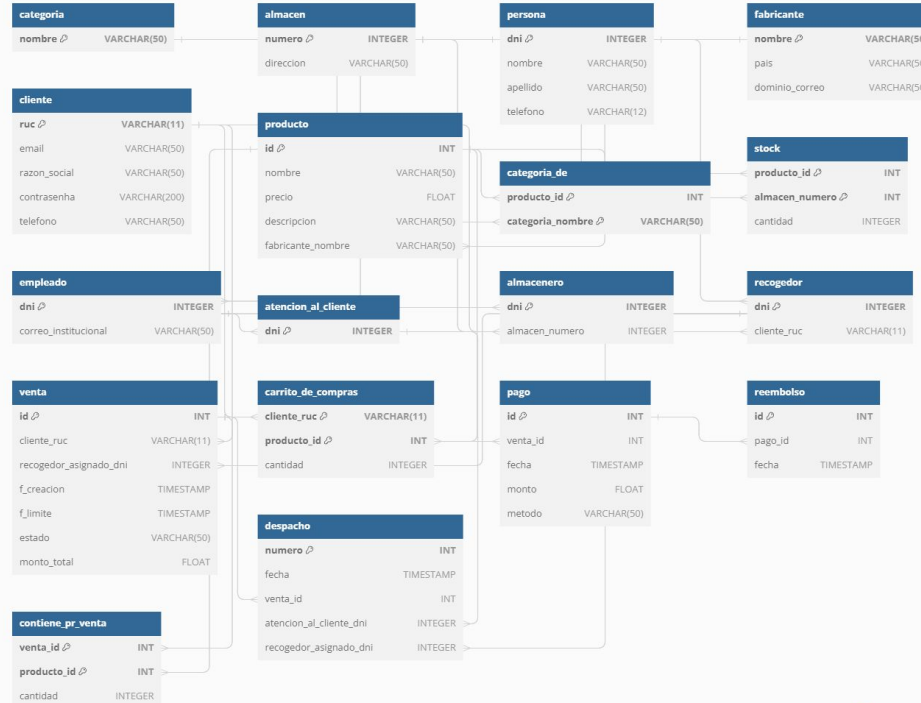
Diagrama de Arquitectura de Solución de Deltron



Este proyecto utiliza cinco contenedores para alojar las siguientes aplicaciones en instancias de Amazon EC2:

- API de Usuarios: Puerto 7145.
- API de Productos: Puerto 7146.
- API de Ventas: Puerto 7147.
- Base de Datos MySQL: Puerto 3306.
- Aplicación Web Frontend: Puerto 8080.

## Diseño de base de datos

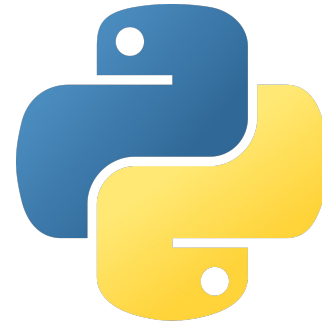


La base de datos utilizada en este proyecto está implementada en MySQL y se encuentra alojada en un contenedor Docker en una instancia de Amazon EC2.

Contiene 18 tablas, las cuales describen un sistema de compra y venta al por mayor de productos electrónicos.

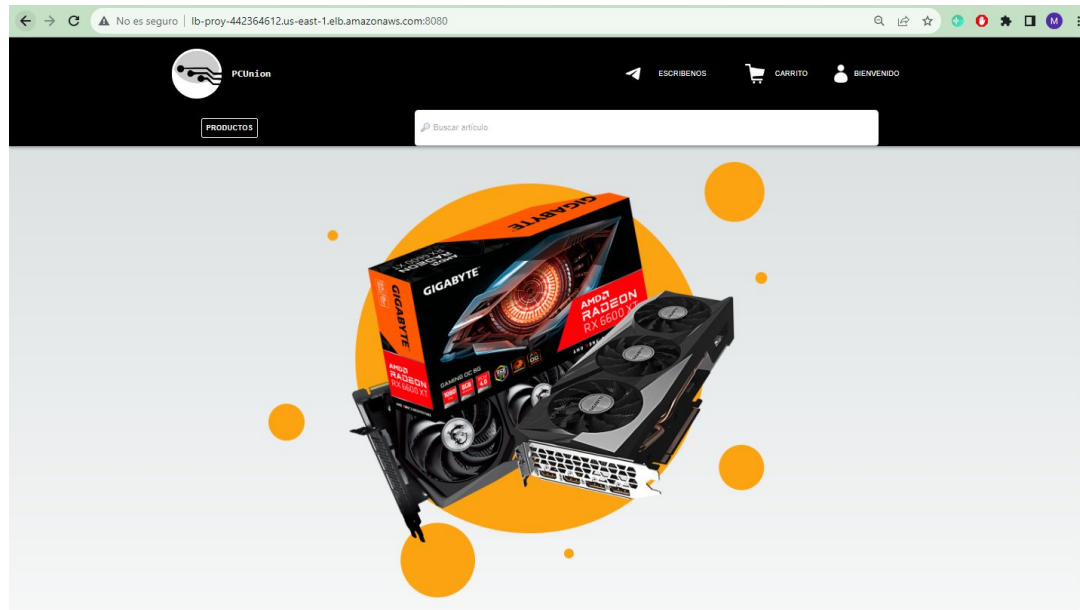
## Funcionalidades de microservicios

- usuarios\_c:
  - Maneja gestiona de datos de clientes. Es necesario para el login respectivo de cada usuario, además de proporcionar datos necesarios relacionados únicamente con los clientes
- productos\_c:
  - Retorna características de productos ya disponibles, o permite añadirlos al catálogo. También permite hacer consultas con filtros para encontrar o realizar búsqueda de productos por TIPO.
- ventas\_c:
  - Permite añadir o obtener ventas, requiere el carrito de compras del respectivo usuario que hace la compra. Para poder generar una petición de compra en la BD.



## Front End

El frontend ha sido construido usando el framework de NEXT. JS, el cual extiende funcionalidades React. Contiene las funcionalidades de logeo de usuarios, visualización de catálogos, filtrado de productos y carritos de compras temporales.



# Demostración

← → ↻ No es seguro | lb-proy-442364612.us-east-1.elb.amazonaws.com:8080/productos/gpu

PCUnion

ESCRIBENOS CARRITO BIENVENIDO


PRODUCTOS

Buscar artículo

**Categorías**


- Todos
- CPU
- GPU
- Memoria RAM
- Almacenamiento
- Refrigeración

**GIGABYTE**



**RTX 2070**  
Descripción faltante...  
Stock: 0  
\$/. 4000

**GIGABYTE**



**RTX 2080**  
Descripción faltante...  
Stock: 0  
\$/. 5000

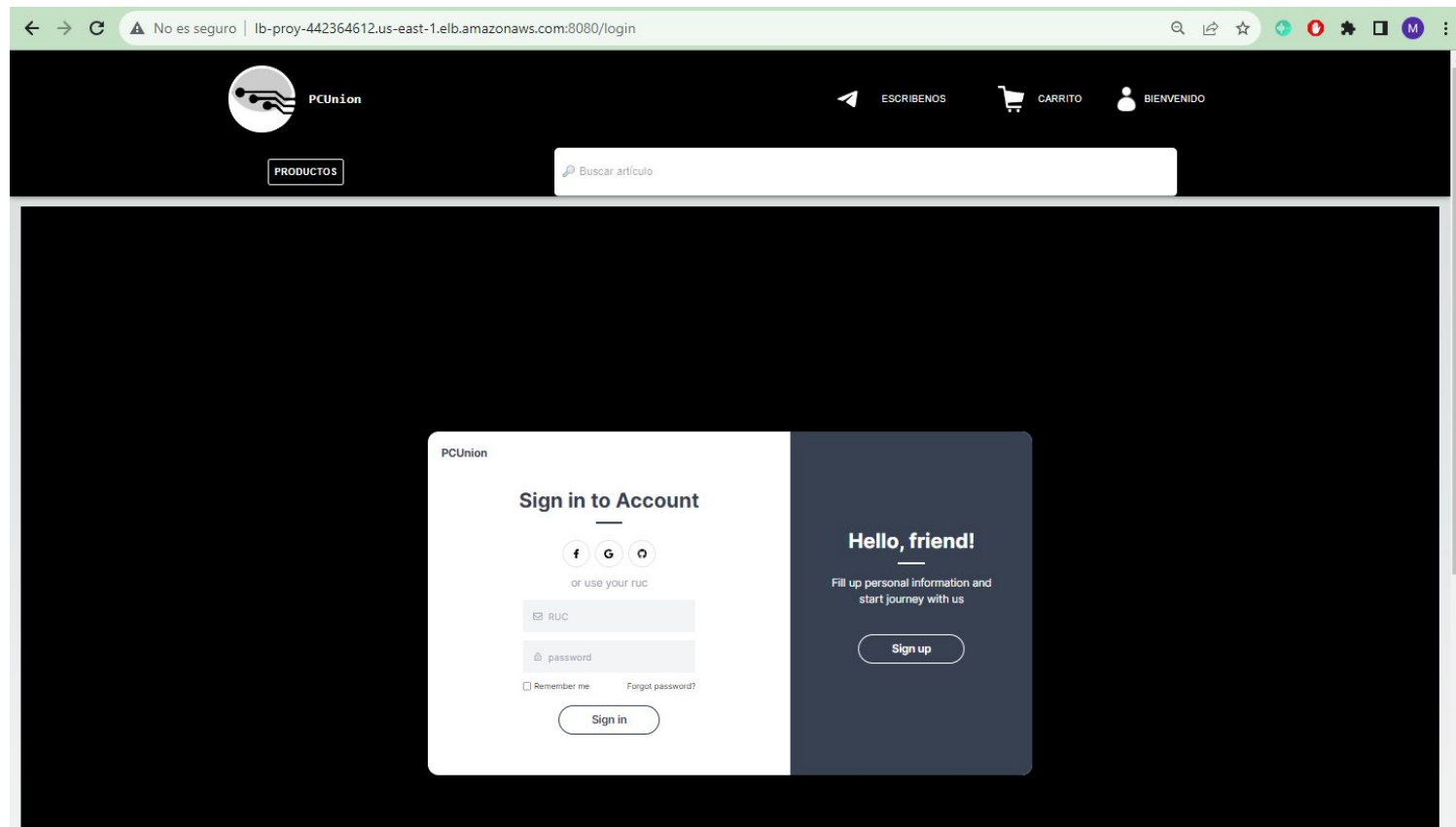
**RTX 2080 Ti**  
Descripción faltante...  
Stock: 0  
\$/. 6000

**GTX 1660**  
Descripción faltante...  
Stock: 0  
\$/. 2000

**GTX 1660 Ti**  
Descripción faltante...  
Stock: 0  
\$/. 3000



# Demostración



The screenshot shows a web browser window with the URL `lb-proy-442364612.us-east-1.elb.amazonaws.com:8080/login`. The page has a dark theme. At the top, there is a navigation bar with the PCUnion logo, a search bar, and links for 'ESCRIBENOS', 'CARRITO', and 'BIENVENIDO'. Below the navigation bar, there is a large dark area with a white login form in the center. The form is titled 'Sign in to Account' and includes fields for 'RUC' and 'password'. There are also links for 'Remember me' and 'Forgot password?'. To the right of the form, there is a dark blue box with the text 'Hello, friend!' and a 'Sign up' button.

PCUnion

PRODUCTOS

Buscar artículo

ESCRIBENOS

CARRITO

BIENVENIDO

PCUnion

### Sign in to Account

or use your ruc

RUC

password

☐ Remember me [Forgot password?](#)

Sign in

Hello, friend!

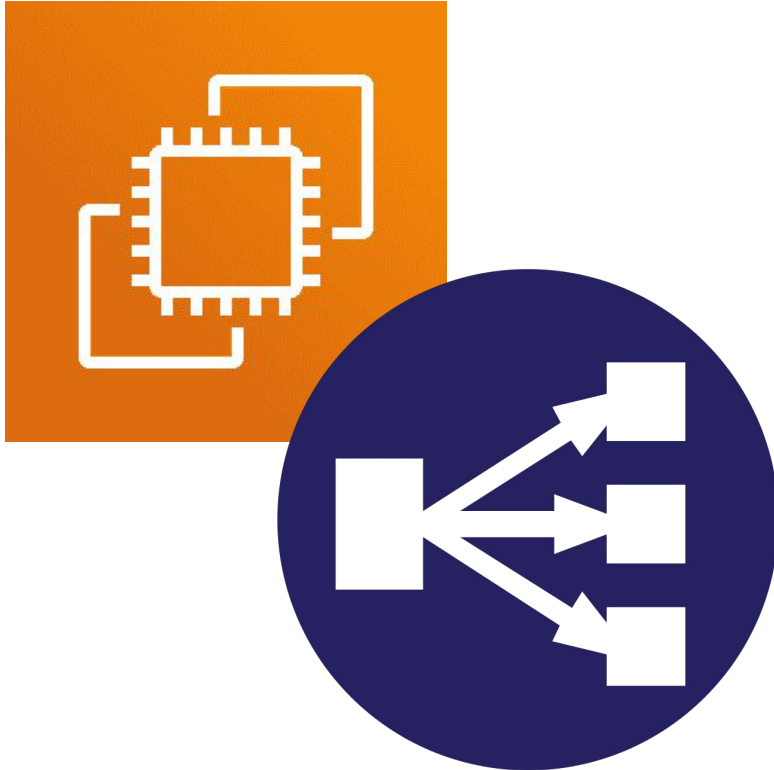
Fill up personal information and start journey with us

Sign up

# 2

## Despliegue en AWS

## Requisitos



- 3 MV:
  - 2 para los microservicios
  - 1 para la BD
- 1 balanceador de carga
- 1 IP elástica
- Archivos necesarios para contenedores (docker-compose)

## Pasos a seguir

- 1) Crear las máquinas virtuales=> P\_MV\_DB, P\_MV\_Prod1, P\_MV\_Prod2
  - a) 1 CPU
  - b) 1 GB RAM
  - c) 10 GB storage

(Asignar las MV de producción con el mismo grupo de seguridad)

*Resultado luego de la creación de las máquinas virtuales:*

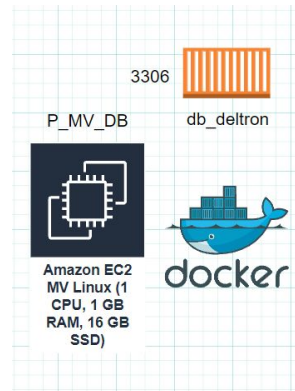
P_VM_Prod2	<a href="#">i-0b6c459624194a548</a>	✓ En ejecución	t2.micro	✓ 2/2 comprobaciones super	Sin alarmas	+	us-east-1c	ec2-54-227-48-5.comp...	54.227.48.5
P_VM_Prod1	<a href="#">i-076762991c8b48dd0</a>	✓ En ejecución	t2.micro	✓ 2/2 comprobaciones super	Sin alarmas	+	us-east-1b	ec2-44-203-138-211.co...	44.203.138.211
P_VM_DB	<a href="#">i-0e04bd8fbc554b8d</a>	✓ En ejecución	t2.micro	✓ 2/2 comprobaciones super	Sin alarmas	+	us-east-1b	ec2-52-7-228-14.comp...	52.7.228.14

## Pasos a seguir

Para la base de datos:

- 1) Crear una IP elástica y asignarla.
- 2) Abrir el puerto 7144 en el grupo de seguridad asignado.
- 3) En la MV crear la carpeta “deltrOn\_db” y poner los contenidos de la carpeta “database” del repositorio de github.
- 4) Dentro de la carpeta ejecutar “docker-compose up”.

**Respositorio Proyecto-deltrOn:** <https://github.com/marcelochincha/Proyecto-parcial-cloud>



## Pasos a seguir

Para el front end y APIs:

- 1) Crear grupos de destino conteniendo a las 2 MV (Prod 1 y 2)

Puertos:

- 8080 (Web)
- 7145 (usuarios)
- 7146 (productos)
- 7147 (ventas)

- 2) Crear un balanceador de carga añadiendo un agente de escucha en cada puerto especificado anteriormente, para posteriormente emparejarlos.

En las 2 MV:

- 3) Crear una carpeta “deltrOn\_app”, y copiar el contenido de “app” del repositorio de github.
- 4) Dentro de la cambiar las credenciales, ip y usuario de la base de datos a la asignada en la parte previa, en cada microservicio (app.py).
- 5) Ejecutar “docker-compose up”

**Respositorio Proyecto-deltrOn:** <https://github.com/marcelochincha/Proyecto-parcial-cloud>

## Observaciones

- Se decidió usar la BD en una máquina virtual EC2 en vez de usar el servicio de RDS, puesto a que este puede llegar a ser más costoso y dar un rendimiento similar. Aunque en el caso de RDS esto se justifica porque la configuraciones son hechas por Amazon.
- Asegurarse que los dockers funcionen con sus respectivos puertos en todas las MV a usar con el balanceador de carga, para evitar problemas por casos donde una falla.



# Gracias