**ARQUITECTURA PROYECTO CONVERTER**

**COMPRESIÓN DE ARCHIVO**

**DESARROLLO DE SOFTWARE EN LA NUBE**

**ESTUDIANTES:**

**JUAN SEBASTIAN BALLESTEROS MUÑOZ**

**CAMILO CORTÉS CAMPO**

**DANIEL FELIPE LEYVA DÍAZ**

**UNIVERSIDAD DE LOS ANDES**

**BOGOTÁ 16 ABRIL 2023**

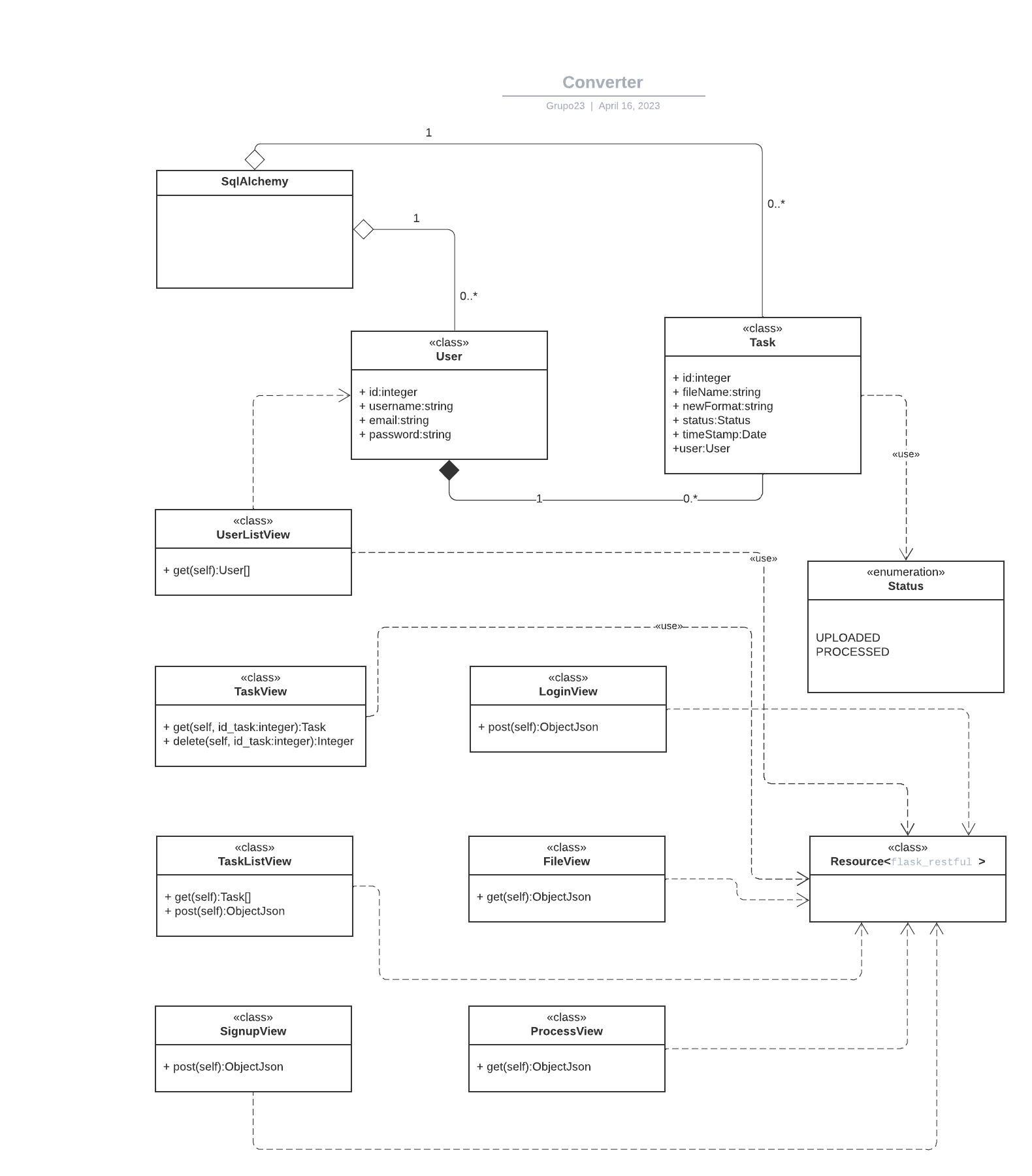
**INTRODUCCIÓN**

Como proyecto de trabajo para cumplir con el requerimiento de la materia se desarrolla una aplicación web que trabaje con servidores web y procesos asíncronos esto con el fin de realizar escenarios de prueba que nos ayuden a medir la capacidad máxima del sistema con esto podríamos dimensionar la capacidad de una aplicación y su infraestructura de soporte tomando como base el plan de pruebas.

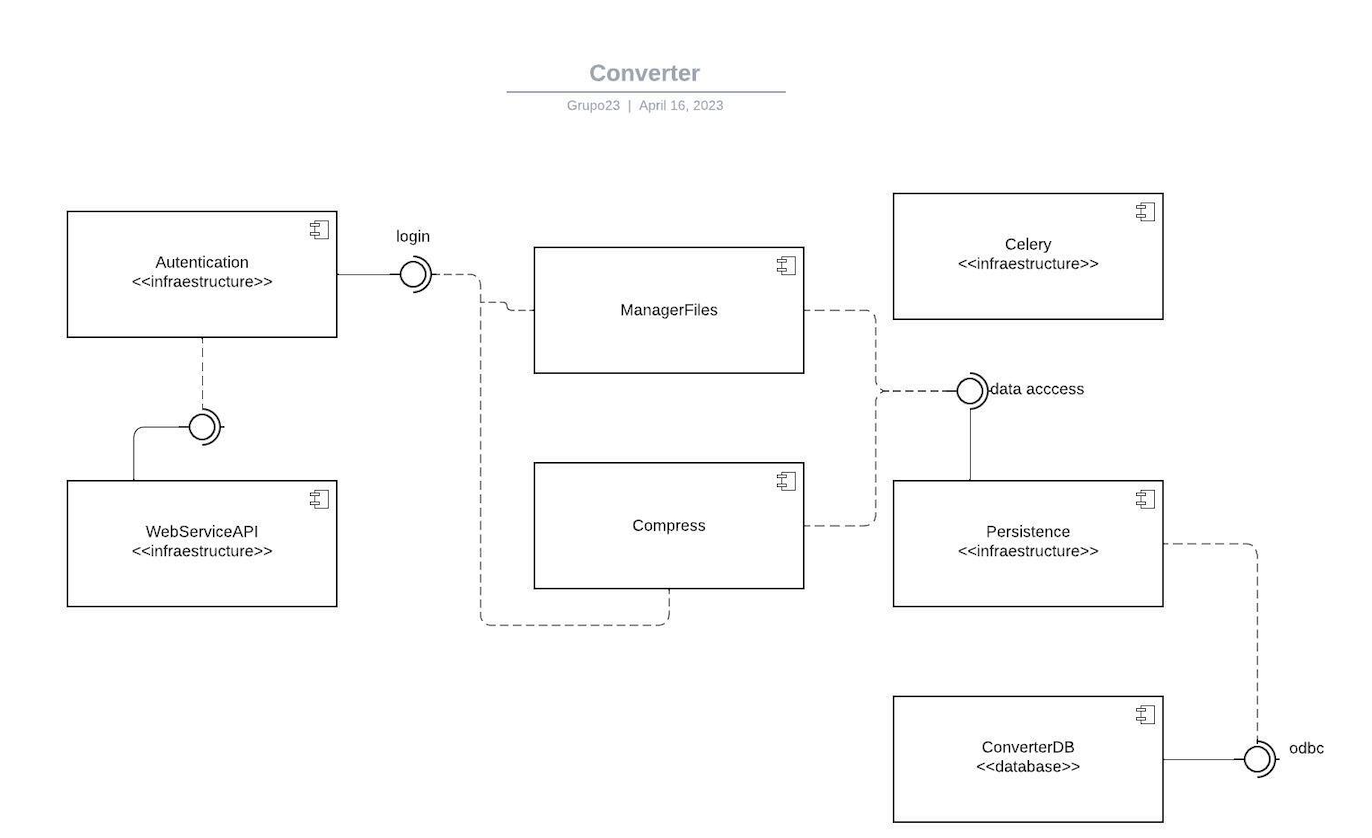
Procedemos a montar una aplicación en lenguaje de programación python con framework FLASK y para el ORM usamos SqlAlchemy, para el procesamiento de tareas en cola usamos CELERY y tenemos los entornos Dockerizados tanto la capa de logica como la capa de persistencia el cual está montado en PostgreSQL. Montamos toda infraestructura en un servicio de AWS que se llama EC2 donde tenemos 2 máquinas virtuales la primera hospeda los contenedores y expone los servicios para ser consumidos y generar el resultado el cual es el archivo comprimido, tenemos la segunda máquina que se encarga de usar la herramienta Apache Benchmark, para todas las pruebas que se necesitan y como ayuda para tener el plan de pruebas totalmente documentado, en la maquina de lógica para el procesamiento de las tareas , usamos Crontab el cual ejecuta cada 5 minutos un endpoint el cual dentro evalúa todas las tareas que se encuentran en proceso UPLOADED esta colección de datos lo que hacemos es proceder a gestionar en cola y hacer la compresión del archivo para posteriormente tener el archivo listo para el usuario final.

Tenemos versionado nuestro proyecto en GitHub tanto el código , como los contenedores de código en archivos de configuración DockerFiles y docker-compose , esto con el fin de poder hacer despliegue más automatizado y con las librerías requeridas, también usamos volúmenes en las máquinas , esto con el fin de que el archivo se cargue en el volumen y la información no se pierda si la máquina se baja , puede ser por que se reinicia la maquina , entonces tenemos los datos con un volumen de persistencia configurado, así mismo la red interna en modo Bridge.

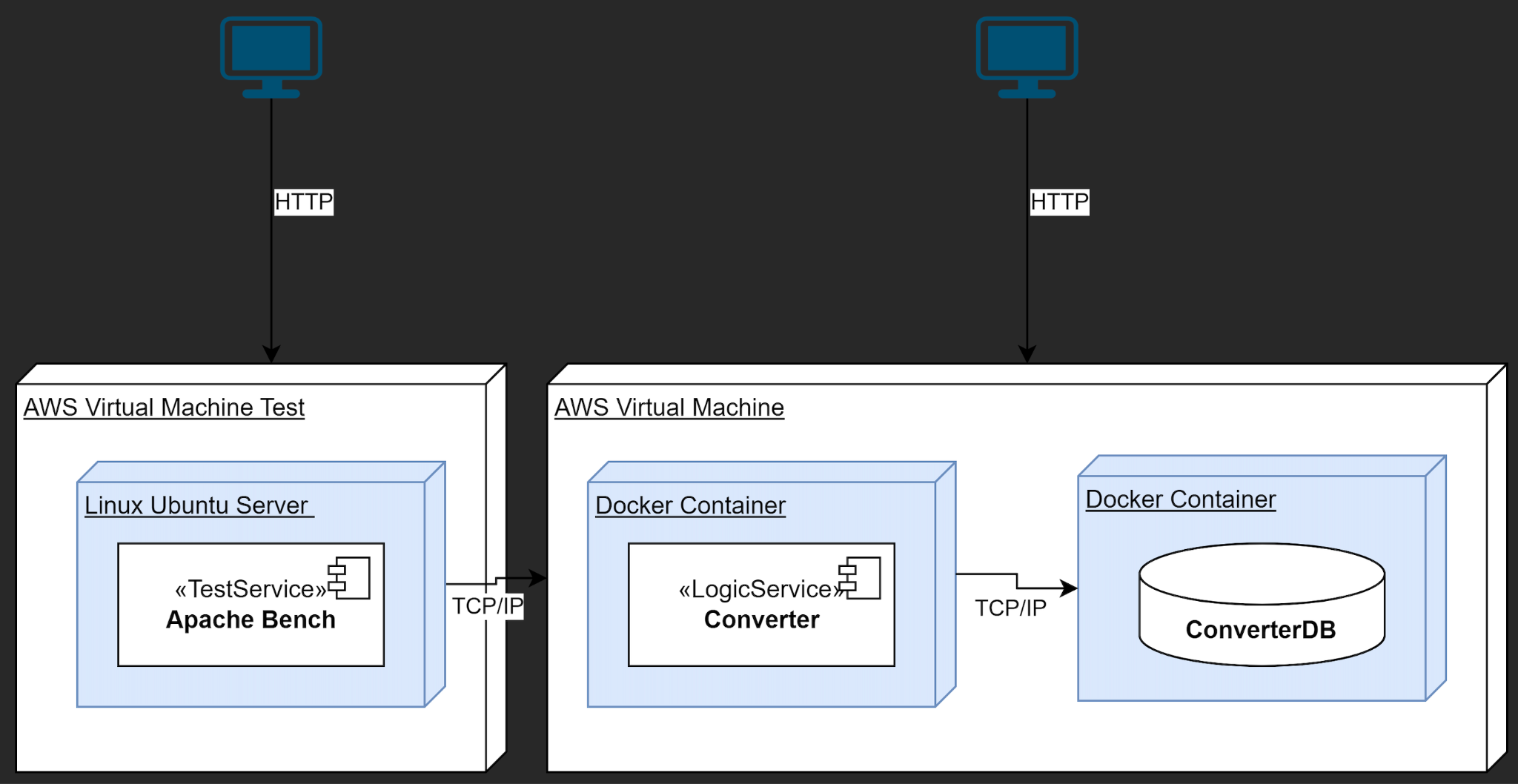
**DIAGRAMA DE CLASES**



**DIAGRAMA DE COMPONENTES**



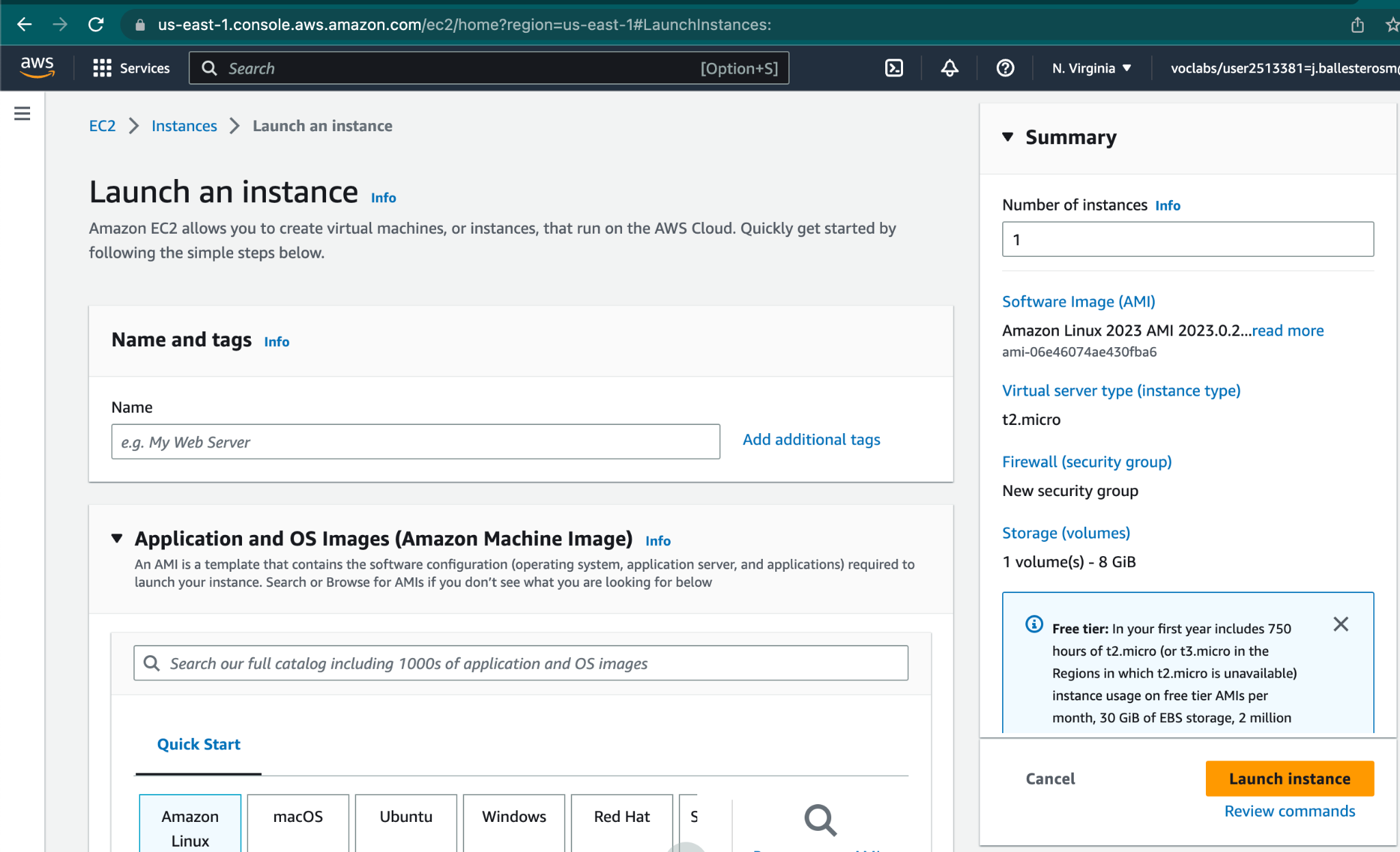
**DIAGRAMA DE INTERFACES**



**PASO A PASO DE CONFIGURACIÓN**

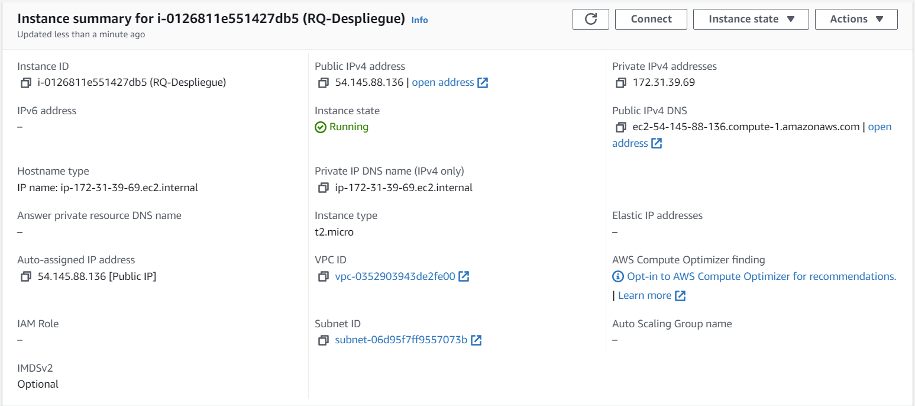
Amazon Web Services (AWS)

Creamos las 2 instancias para este proyecto en ubuntu con las características solicitadas por los requerimientos del instructor.

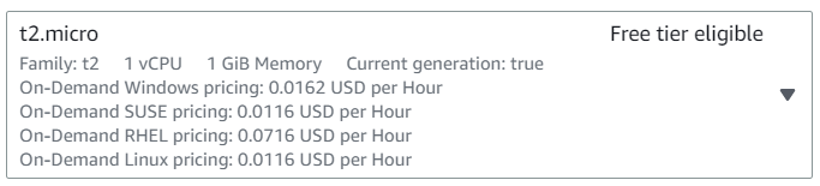


**MAQUINA DESPLIEGUE (AMBIENTE 1 AWS):**

* 20 GB Almacenamiento
* 1 GB RAM
* 1 CPU Virtual
* Distribución Ubuntu Server 20.04 LTS



* Tipo de instancia t2.micro



**Sistema Operativo**

Utilizamos el sistema operativo Ubuntu Server 20.04 LTS

Se realiza la creación de la subred para habilitar la comunicación con el ambiente de pruebas:

Se configura el grupo de seguridad para el mismo objetivo

Instalación Herramientas

Docker

• sudo apt install docker-ce

Docker-Compose

• sudo curl -L "https://github.com/docker/compose/releases/download/1.26.0/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

Se realiza la configuración en la aplicación para trabajar con Docker-Compose el despliegue de la aplicación:

Por último, se realiza la configuración para ejecutar el Docker-compose en el archivo Docker-compose.yml

Ejecución del Docker-compose

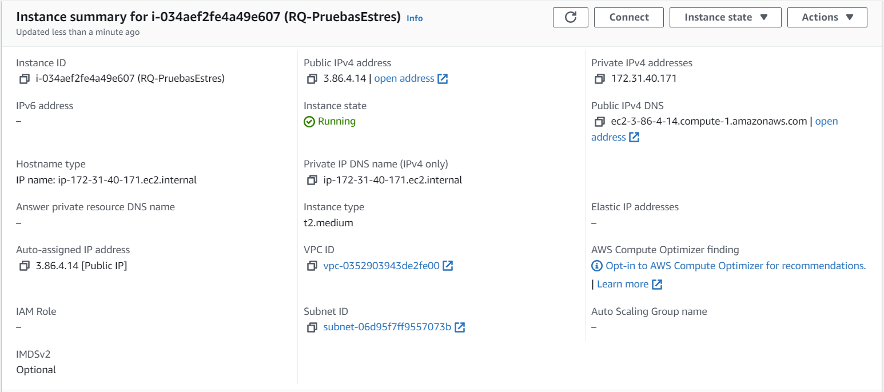
En la ruta raíz del proyecto ejecutamos las sentencia para subir el docker-compose

• sudo docker-compose up

**MAQUINA PRUEBAS (AMBIENTE 1 AWS):**

* 20 GB Almacenamiento
* 1 GB RAM
* 2 CPU Virtual
* Distribución Ubuntu Server 20.04 LTS

Creación Máquina virtual



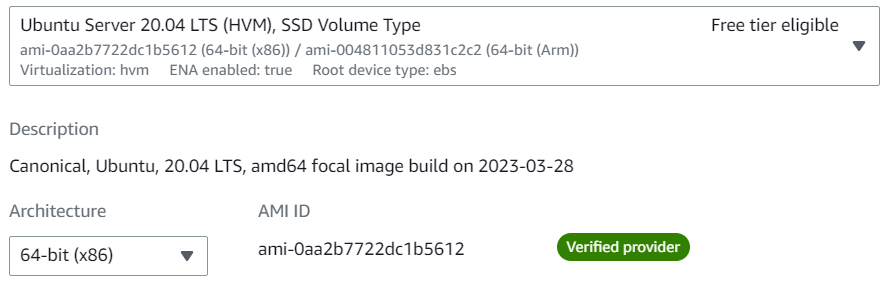
Tipo de instancia

Para crear la máquina virtual con las características mencionadas anteriormente se utiliza una instancia EC2 de Amazon web servicies de tipo t2.medium:



Sistema Operativo

Utilizamos el sistema operativo Ubuntu Server 20.04 LTS



Se realiza la creación de la subred para habilitar la comunicación con el ambiente de pruebas:

Se configura el grupo de seguridad para el mismo objetivo

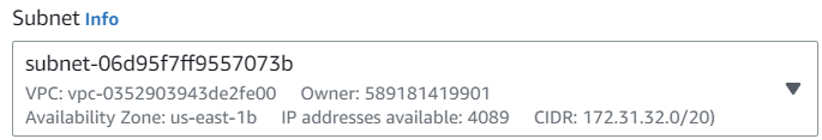
Instalación de Herramientas

Apache Bench

• sudo apt install apache2-utils

gnuplot

• sudo apt-get install gnuplot



Se configura el grupo de seguridad para el mismo objetivo

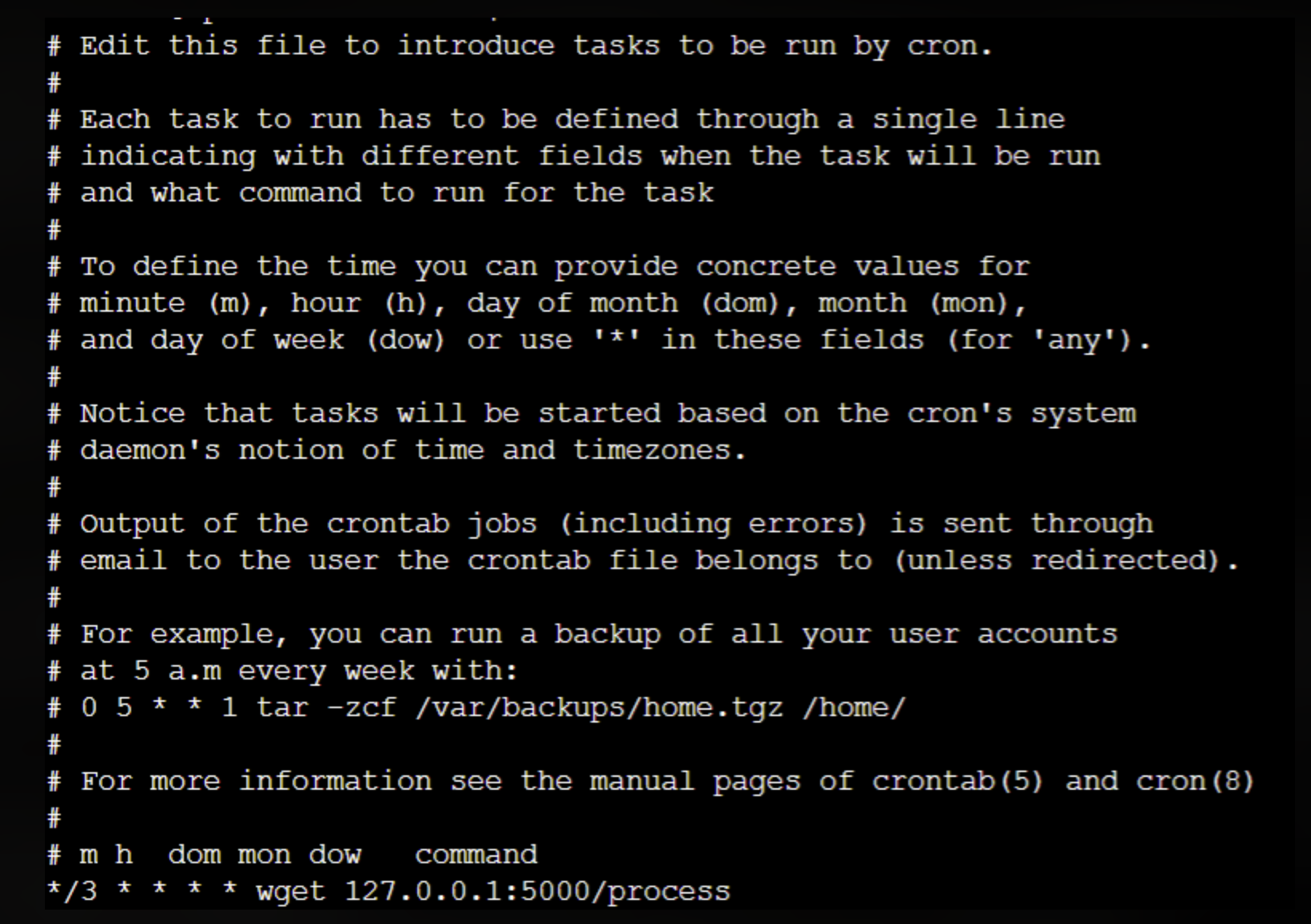
Instalación de Herramientas

Apache Bench

• sudo apt install apache2-utils

gnuplot

• sudo apt-get install gnuplot

**CRONTAB**

Este crontab corre cada 5 minutos verificando que tenga task.