

Universidad San Carlos de Guatemala  
Facultad de Ingeniería  
Ingeniería en Ciencias y Sistemas  
Sistemas Operativos 1  
Ing. Sergio Méndez  
Junio 2020

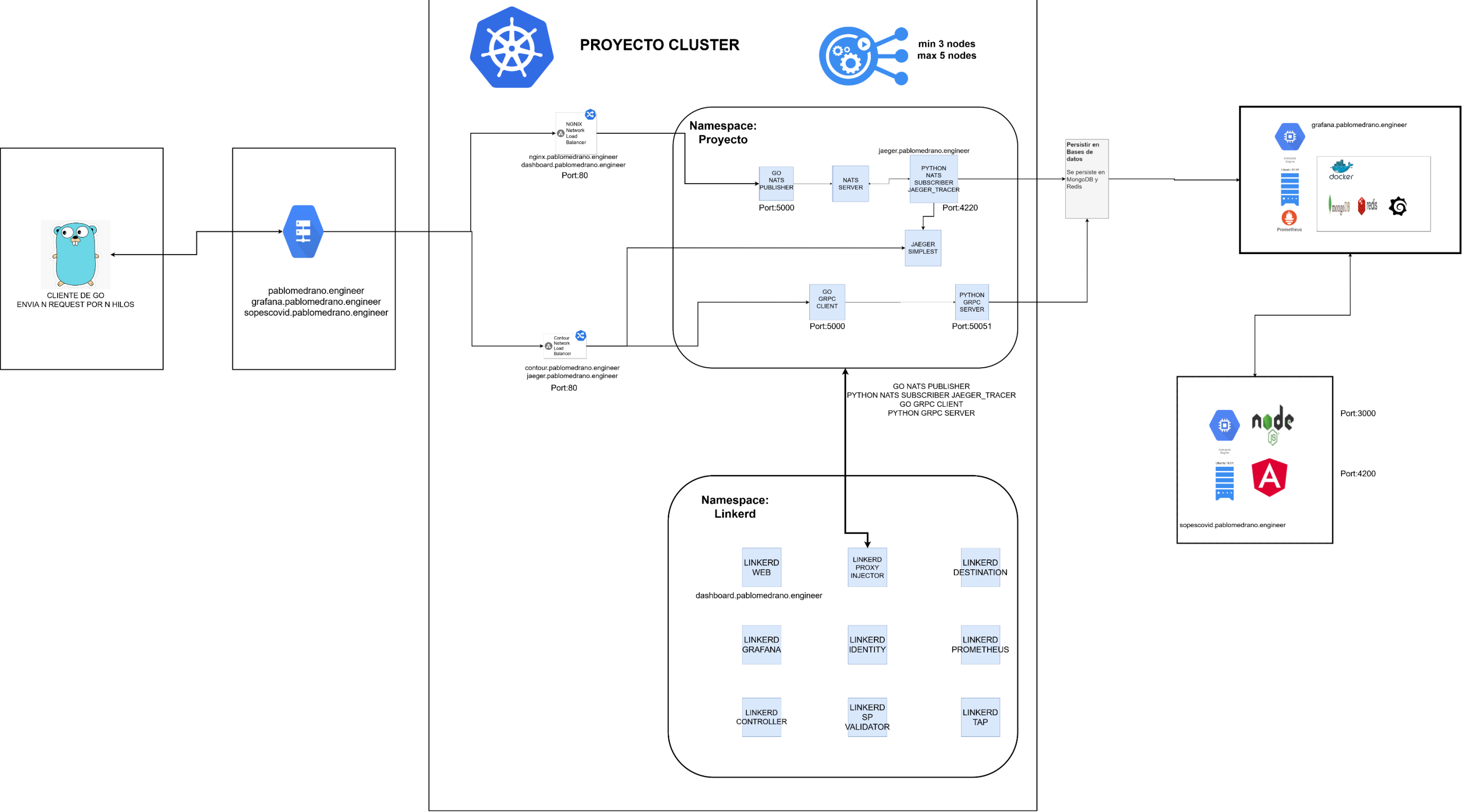


## MANUAL SEGUNDO PROYECTO

Pablo Medrano 201222552

Dennis Castro 201213146

DIAGRAMA DE ARQUITECTURA DEL PROYECTO



## Implementación

El proyecto ha sido implementado en la plataforma Google Cloud, se han empleado las tecnologías de computación en la nube, Kubernetes, Docker Containers. Cloud DNS. Se ha creado un cluster con 3 nodos y auto escalable hasta un máximo de 5 nodos.

Dentro del cluster se han creado dos namespaces:

1. **Proyecto**
2. **Linkerd**

**Namespace Proyecto:** En este se encuentran todos los pods funcionales los mismos se pueden ver en el diagrama del proyecto, se especifica su puerto donde reciben y envían las peticiones.

**Namespace Linkerd:** Este ha sido creado exclusivamente para el uso de Linkerd el cual incluye todos los pods necesarios para una integración completa.

**Nota:** para la inyección de Mesh Service en los pods del namespace proyecto debe ser de forma individual de lo contrario se corre el riesgo de provocar fallos en el cluster y los servicios.

Se ha comprado un domino el cual es: **pablomedrano.engineer**

los subdominios registrados son:

balanceador de carga y reverse proxy nginx:  
<http://nginx.pablomedrano.engineer>

balanceador de carga y reverse proxy contour  
<http://contour.pablomedrano.engineer>

Dashboard para Linkerd

<http://dashboard.pablomedrano.engineer>

dashboard para Jaeger

<http://jaeger.pablomedrano.engineer>

dashboard para grafana del servidor de bases de datos

<http://grafana.pablomedrano.engineer>

sitio web del proyecto

<http://sopescovid.pablomedrano.engineer>

Los lenguajes de programación utilizados para resolver el proyecto han sido Golang y Python en la parte de servicios.

Para mensajería entre aplicaciones se han implementado: NATs Y gRPC

Las bases de datos han sido implementadas: MongoDB y Redis.

Lenguajes utilizados En el portal web en Backend y Frontend respectivamente: NodeJS y Angular.

Para la telemetría y monitoreo en el clúster se han implementado Linkerd, Prometheus, Grafana y Jaeger

Para monitoreo en el servidor de bases de datos se ha implementado: Prometheus y Grafana.

**El proyecto consta de 5 fases:**

- 1. Envío de Datos,**
- 2. Balanceadores de carga**
- 3. Mensajería entre aplicaciones**
- 4. Bases de datos**
- 5. Sitio Web**

El Proyecto incluye también un apartado de Telemetría y Monitoreo

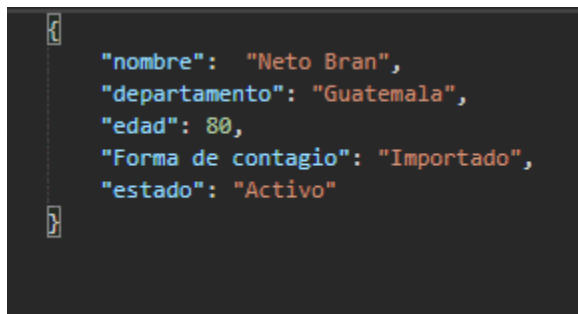
Dicho aparatado ha sido desarrollado con las tecnologías mencionadas anteriormente.

## 1. Envió de datos

Para la primera parte se ha desarrollado un programa en Golang se le especifica a cuál de los balanceadores de carga se le enviarán las peticiones y número de hilos (solicitudes en paralelo) que se enviarán, los datos están en un archivo de entrada, los balanceadores de carga han sido implementados en:

- **Contour**
- **Nginx**

El archivo con los datos tendrá el siguiente formato:

A screenshot of a code editor showing a JSON object. The object has five key-value pairs: "nombre" with value "Neto Bran", "departamento" with value "Guatemala", "edad" with value 80, "Forma de contagio" with value "Importado", and "estado" with value "Activo". The code is color-coded: strings are orange, numbers are green, and keys are blue. The JSON is enclosed in curly braces with double quotes around the string values.

```
{  
  "nombre": "Neto Bran",  
  "departamento": "Guatemala",  
  "edad": 80,  
  "Forma de contagio": "Importado",  
  "estado": "Activo"  
}
```

El tendrá N cantidad de datos (para la calificación será proporcionado por el auxiliar).

Cuando se ejecute el programa de Go se pregunta lo siguiente:

Url del balanceador de carga que se desea enviar

Cantidad de hilos que se desean para enviar

Cantidad de solicitudes que tiene el archivo (la cantidad de casos que se desean enviar)

Ruta del archivo que se desea cargar

## 2. Balanceadores de carga

Desde esta fase hasta la 5 se ha implementado en Google Cloud.

La segunda parte empieza por instalar kubernetes la cual será la base de las aplicaciones.

### Balanceadores de carga (Nginx y Contour):

Ambos realizan la misma funcionalidad sin embargo los balanceadores de carga son distintos, los balanceadores de carga se han instalado contenedores que tendrán cada uno un servidor web corriendo sobre go. Dichos clientes exponen el puerto 5000

Su funcionalidad principal está dada bajo el servicio de mensajería que utiliza su línea de distribución. Para la línea de nginx se ha implementado GO como un Publisher con un servidor NATs.

Para la línea de contour se ha implementado GO como un cliente de un servidor gRPC en Python.

Para ello se han creado dos deployments para la línea de nginx

El cliente Go es el deployment <<miapp>>

Para la línea de contour el cliente go es el deployment <<miapp-golan>>

Ambos han sido expuestos como servicios http siendo respectivamente:

[nginx.pablomedrano.engineer](#) y [contour.pablomedrano.engineer](#)

### 3. Mensajería entre aplicaciones

Se obtiene la información que se envió desde el cliente local y se manda por medio dos herramientas:

**NATS.io:** Es un sistema de mensajería de código abierto simple, seguro y de alto rendimiento para aplicaciones nativas de la nube, mensajería IoT y arquitecturas de microservicios. este servidor utiliza el puerto 4220 y se expone como el servicio <<nats>>

**gRPC:** Es un framework moderno de llamada a procedimiento remoto (RPC) de código abierto que puede ejecutarse en cualquier lugar. Permite que las aplicaciones cliente y servidor se comuniquen de manera transparente y facilita la creación de sistemas conectados. El servidor de gRPC utiliza el puerto 50051

La información es enviada a dos deployments que corren una aplicación en Python, estas aplicaciones se encargarán de almacenar los datos finalmente en las dos bases de datos que se estarán manejando.

Para la línea de nginx se ha implementado un Subscriber en Python, al mismo tiempo dicha aplicación lleva el trazado de las operaciones y envía la información a el pod de Jaeger.

Para la línea de contour, se ha implementado un servidor gRPC en Python.

### 4. Bases de datos

Se han implementado las bases de datos no relaciones para mejorar la velocidad y rendimiento de la aplicación, dichas bases de datos han sido implementadas con Docker Containers sobre una Máquina Virtual en Google cloud con el sistema operativo Ubuntu:18.04

las bases de datos son:

**MongoDB:** Es un sistema de base de datos NoSQL, orientado a documentos y de código abierto. En lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. El container expone el puerto 27017

**Redis:** Es un almacén de datos en memoria de código abierto (con licencia BSD), utilizado como base de datos, caché y agente de mensajes. Admite estructuras de datos como cadenas, hashes, listas, conjuntos, conjuntos ordenados con consultas de rango, mapas de bits, hiperloglogs, índices geoespaciales con consultas y flujos de radio. El container expone el puerto 6379

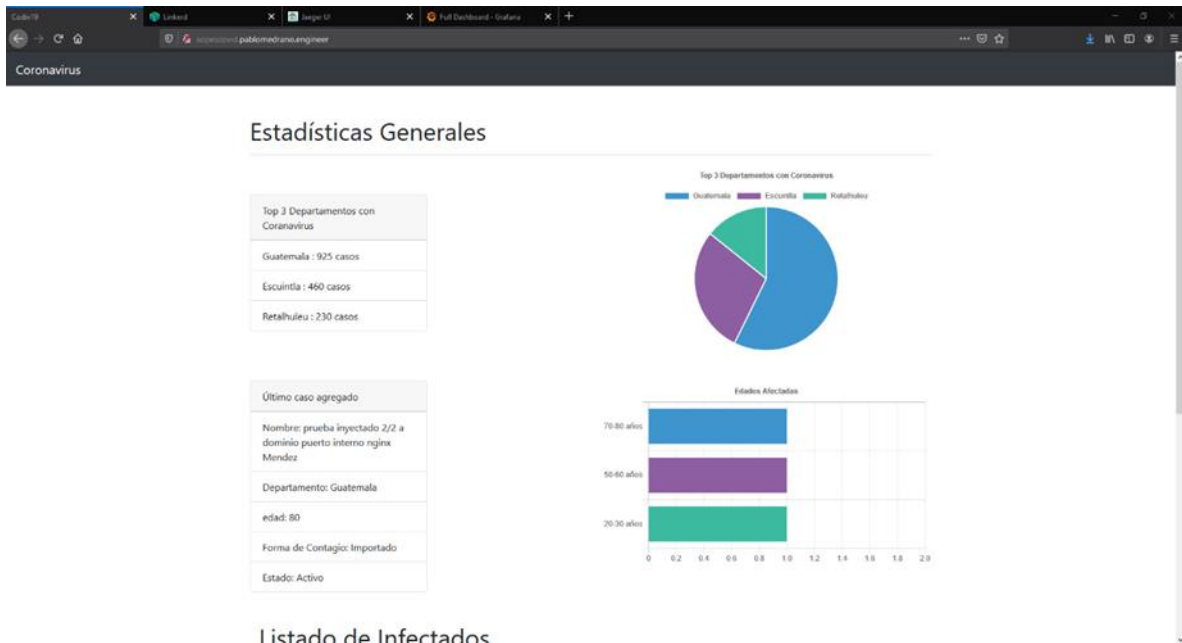
## 5. Página Web

Por último, se ha creado una página web corriendo sobre un servidor Ubuntu:18.04 con nodejs y frontend Angular, dentro de la página la cual se tiene una temática de coronavirus se tiene lo siguiente

### Apartados:

- Muestra una tabla con todos los datos (MongoDB).
- Muestra el top tres de departamentos con más casos (MongoDB).
- Gráfica de Pie con todos los departamentos afectados (MongoDB).
- El último caso agregado (el último del archivo) (Redis).
- Una gráfica de barras que muestra la cantidad de
- afectados por rango de edades, ej. 0-10 años 5 casos (Redis).

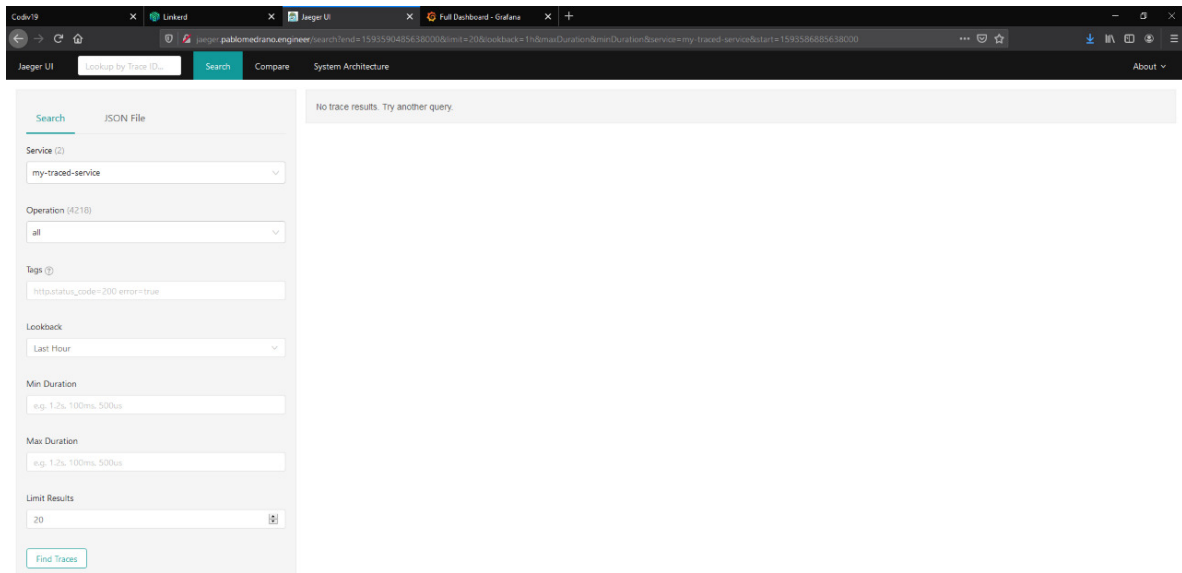




## MONITOREO

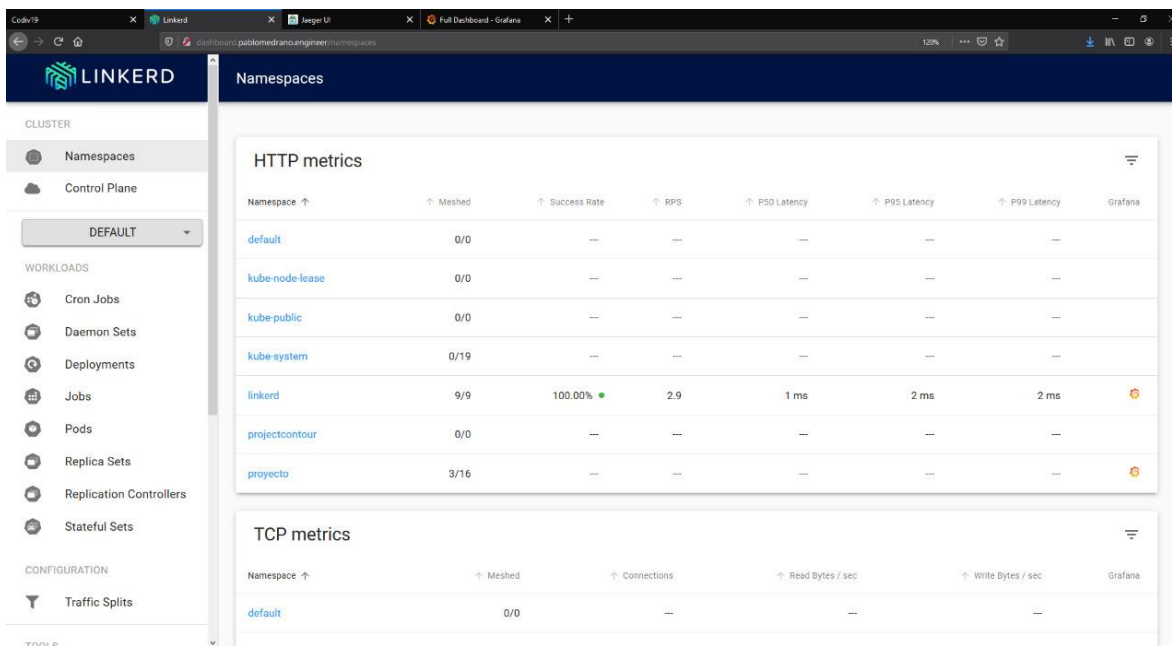
### Jaeger

Se ha implementado jaeger tracing en todos los contenedores de la línea superior para poder monitorear todas las acciones que suceden en estos.



## Linkerd

Se ha implementado Linkerd en los contenedores de la línea inferior y superior, esta es una herramienta que trabaja en conjunto con kubernetes para el monitoreo de los contenedores.

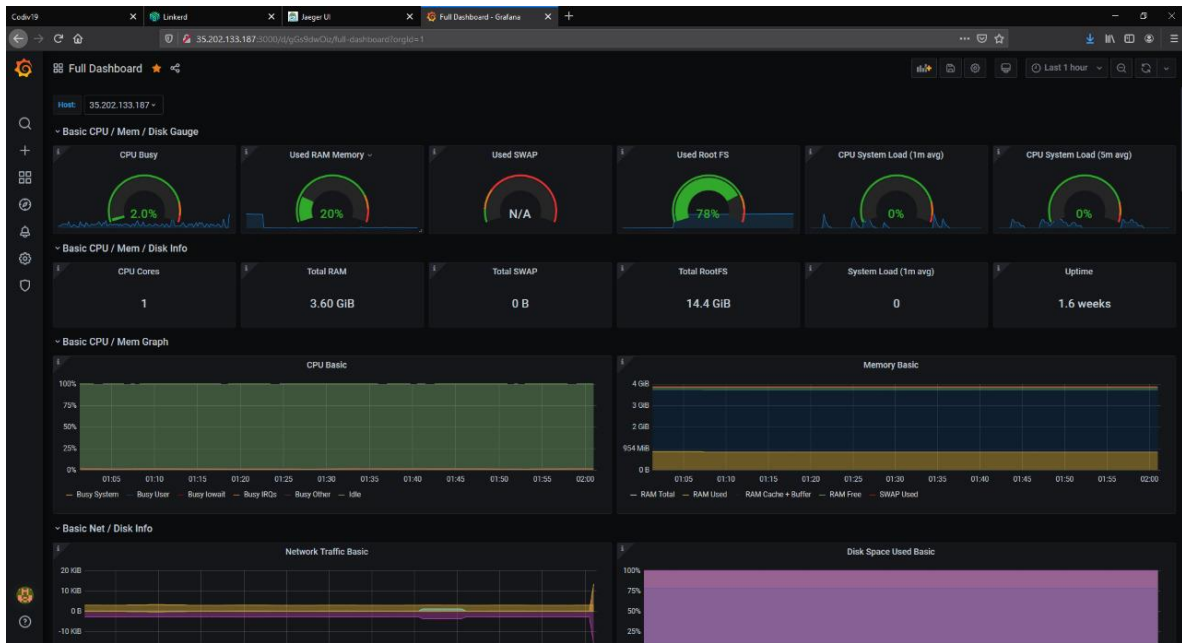


## Prometheus

Se ha instalado el software de Prometheus para el monitoreo de las máquinas virtuales que contienen las bases de datos.

## Grafana

Se ha instalado el software Grafana para mostrar *dashboards* con los datos obtenidos con Prometheus en el servidor de las bases de datos



Arquitectura proporcionada en el enunciado del proyecto:

