Apuntes Bases de Datos II - Clase 14/02/2023

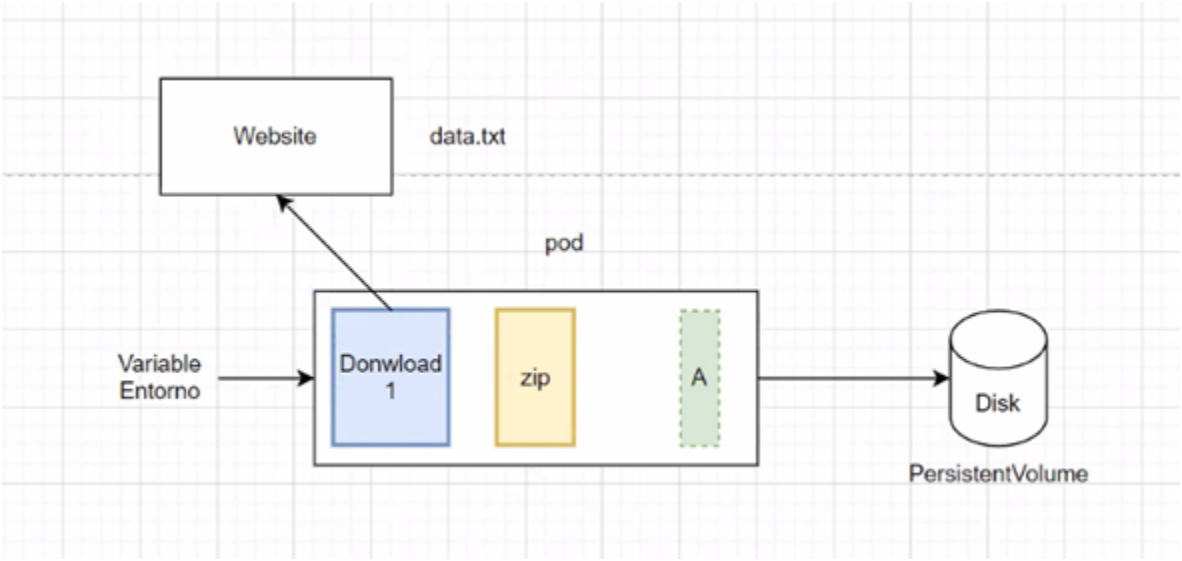
Gerald Núñez Chavarría - 2021023226

Profesor: Nereo Campos Araya.

En la clase del día martes 14 de Febrero se vieron un conjuntos de componentes que son útiles para la realización del proyecto opcional del curso.

1. Un ***POD*** es el objeto más básico y pequeño de Kubernets, representa una instancia única de un proceso de ejecución, también se puede decir que un POD es un conjunto de uno o varios contenedores. El ***contenedor*** es, en palabras sencillas, un bloque que ejecuta una tarea. Es importante qué los contenedores dentro de un POD realicen tareas especificas, en lugar de tener un solo contenedor con muchas tareas.

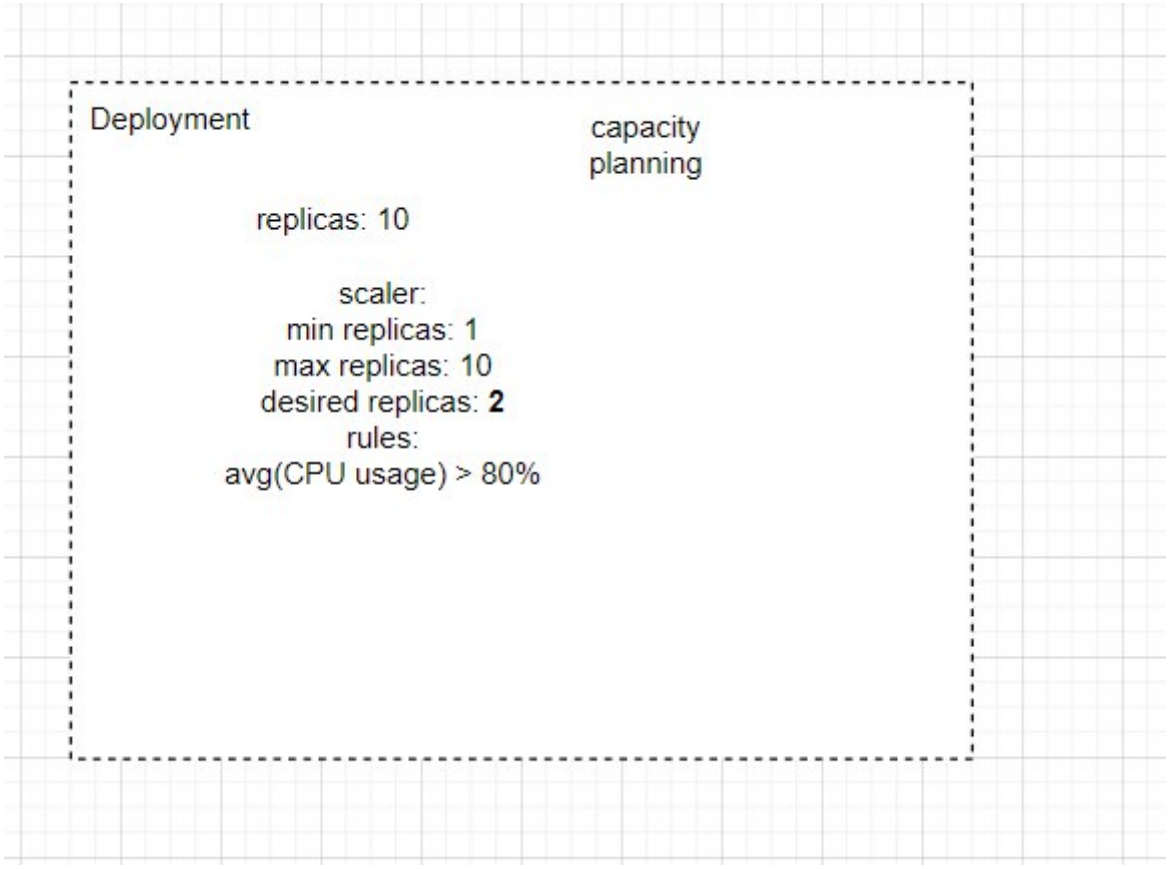
   Ahora se puede observar un ejemplo dónde se tiene un POD, cuya tarea es bajar la información de un sitio web y llevarla al disco, se puede ver como este POD tiene tres contenedores realizando tareas especificas dentro de él.



2. ***Deployment*** es una unidad de alto nivel que se puede gestionar en Kubernets, permite entre sus funciones, tener una cantidad "X" de replicas de un POD, realizar backup para recuperar a uno de sus POD cuando este se cae.

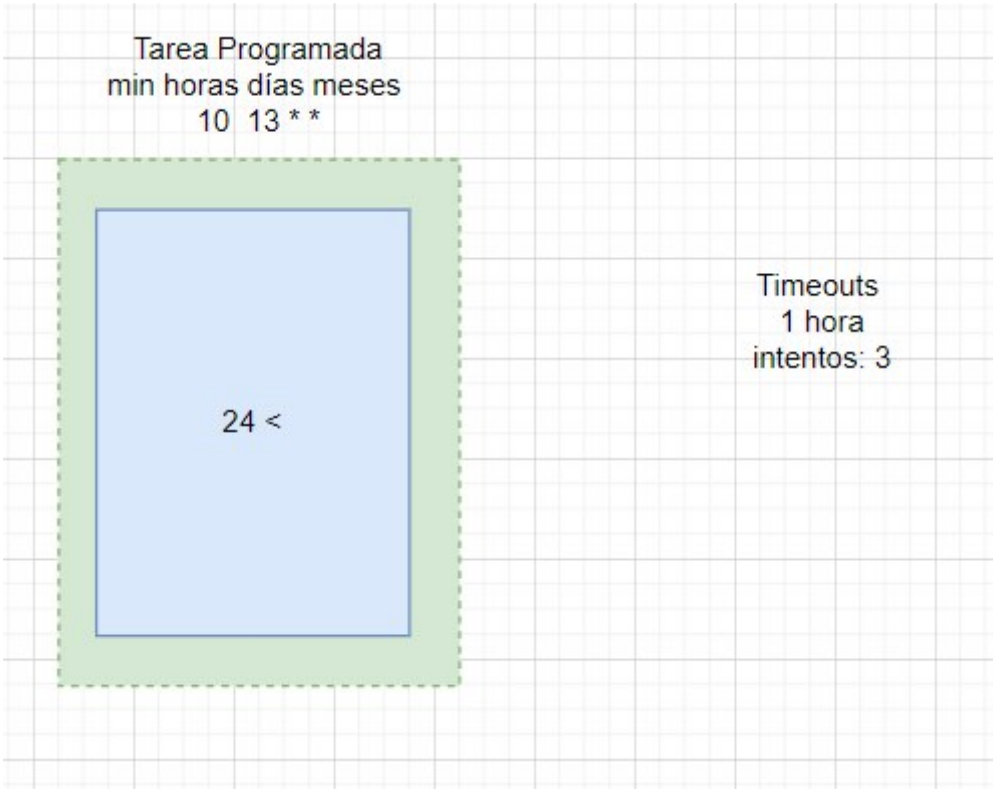
   Es ideal que un deployment cuente con un ***capacity planing***, consiste en aplicarle un scaler a un sistema para que funcione automático, permitiendo que el sistema sea flexible de acuerdo a la demanda.

   En el ejemplo a continuación, se declara un número mínimo de réplicas, un número máximo, una cantidad inicial, se designa el estudio de dos réplicas, y se pone la regla de que cuando el promedio de uso del CPU de esa replicas estudiadas sea mayor a 80%, entonces se deben agregar dos replicas nuevas, así mismo, si el sistema baja la demanda de CPU, entonces se quitan dos replicas, permitiendo un sistema elástico que funciona de acuerdo a la demanda.



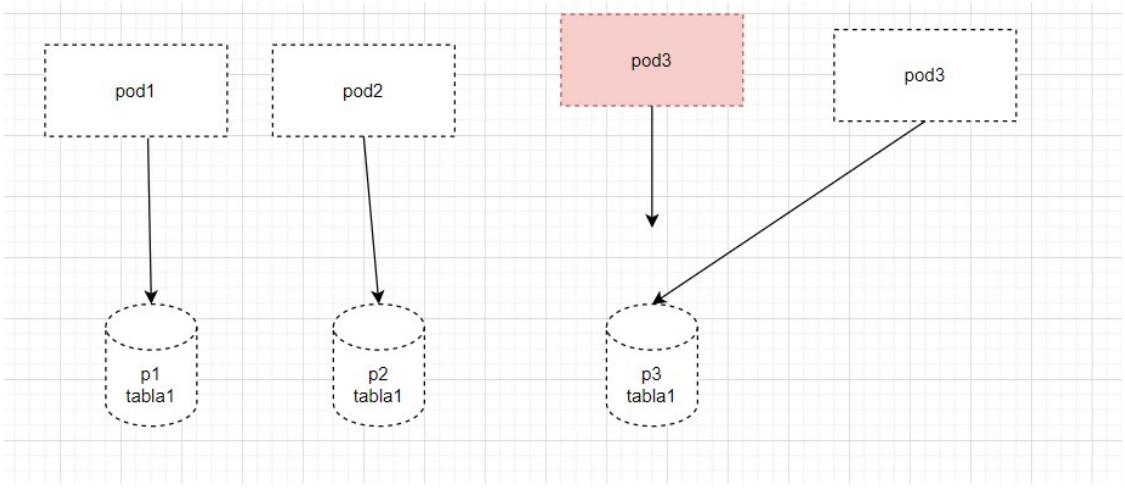
3. Como lo ideal es automatizar un sistema, otro instrumento importante es el ***Cronjob***, un concepto del sistema Unix que básicamente es una tarea programada. La idea es que cuando un POD tiene un tiempo de ejecución menor a 24 horas, se define un Cronjob para ejecutarlo en los minutos, horas, días, meses, años especificados. Al cronjob se le puede definir el tiempo de duración que debería tener (timeout) y también definirle el número de intentos de ejecución.

   El ejemplo anterior se define un cronjob que se ejecuta a los 13:00 horas en el minuto 10, todos los días, se tiene un timeout de 1 hora y máximo 3 intentos. Sin en los 3 intentos se tarda más de una hora en la ejecución del POD, quiere decir que algo anda mal en el POD.

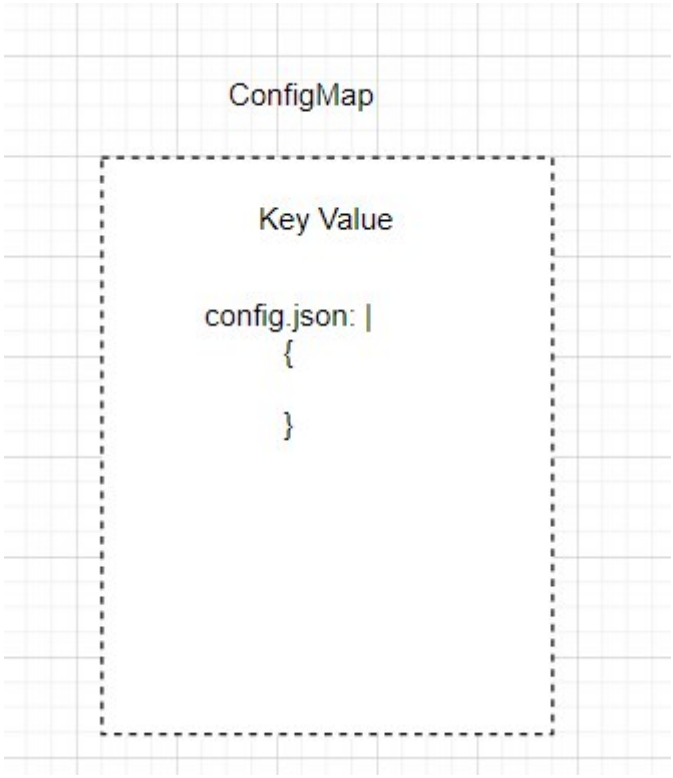


4. Un ***StatefulSet*** es básicamente un deployment que escribe las conexiones de disco que tienen los PODs. Esto garantiza que cuando se aumenta o disminuye las réplicas, los discos se mantienen igual.

   Para entender su funcionamiento, observe el ejemplo que viene a continuación, dónde se tiene una tabla de una base de datos a la que se accede, pero esta tiene tres particiones p1, p2 y p3, donde se conectan el POD1, POD2 y POD3 respectivamente, sin embargo, si el POD3 muere (el de color rosado significa que murió esa réplica de POD), lo que hace statefulset es levantar una réplica de ese POD con el mismo nombre del que murió (POD3), esto genera POD que trae los mismos valores en disco del que murió, con esto se dicen que los discos juegan igual.

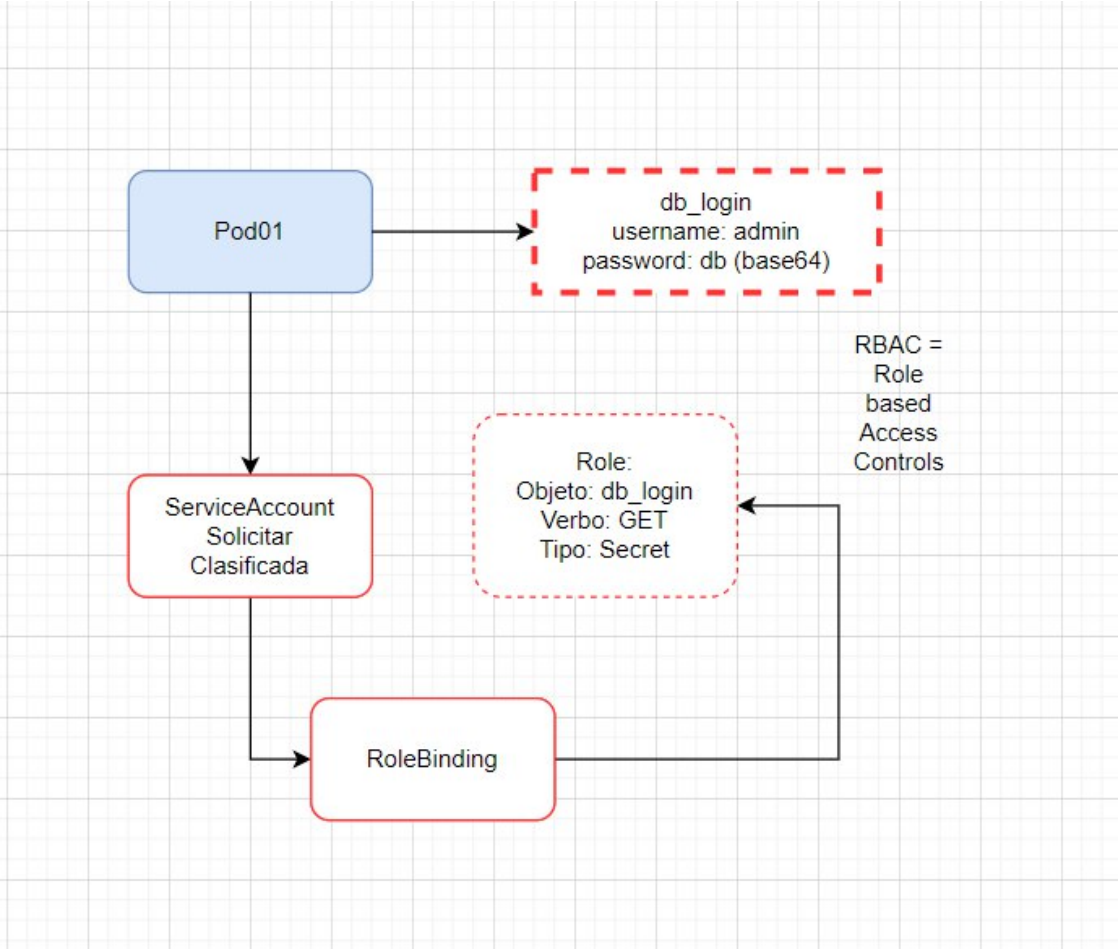


5. El ***configmap*** es un tipo de archivo que permite guardar configuraciones de cualquier tipo, por ejemplo en un lenguaje como jason. Se le puede asignar configmaps a un POD y este puede acceder a manipular la información y utilizarla para lo que necesite.

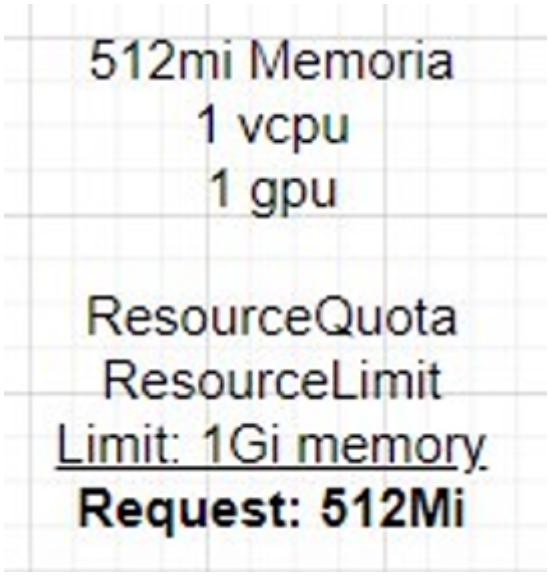


6. Los **secrets** son semejantes a los configmap pero su información esta encriptada, se cuenta con un nombre de usuario y una contraseña. Un POD utiliza un ***ServiceAccount*** que permite el acceso a información clasificada, este se conecta a un ***RoleBinding*** qué es simplemente un conector entre el ServiceAccount y el ***Role***, que tiene un objeto (nombre del objeto a acceder), un verbo (la acción que se quiere hacer en el objeto) y un tipo (el tipo del objeto), y ya con este Role podemos acceder por ejemplo a un objeto de tipo secret. Cuando un POD se conecta a un ServiceAccount, por herencia obtiene el derecho de leer los secrets conectados a este ServiceAccount.

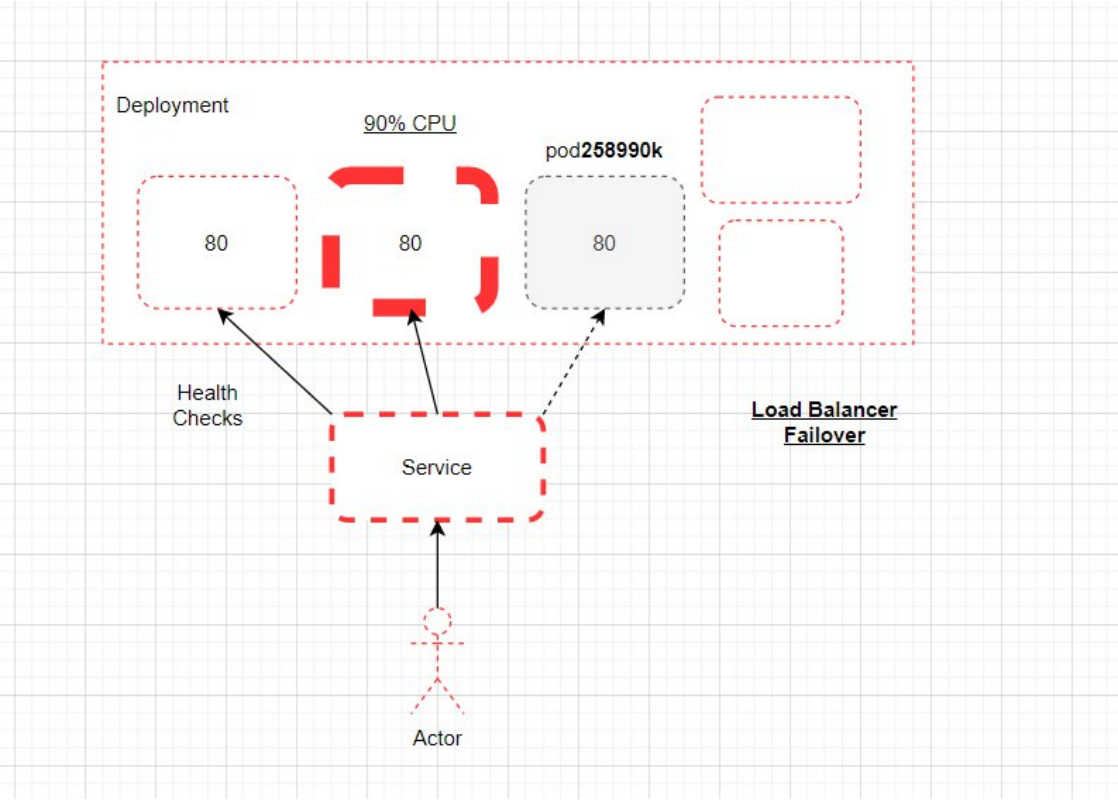
   Ejemplo:



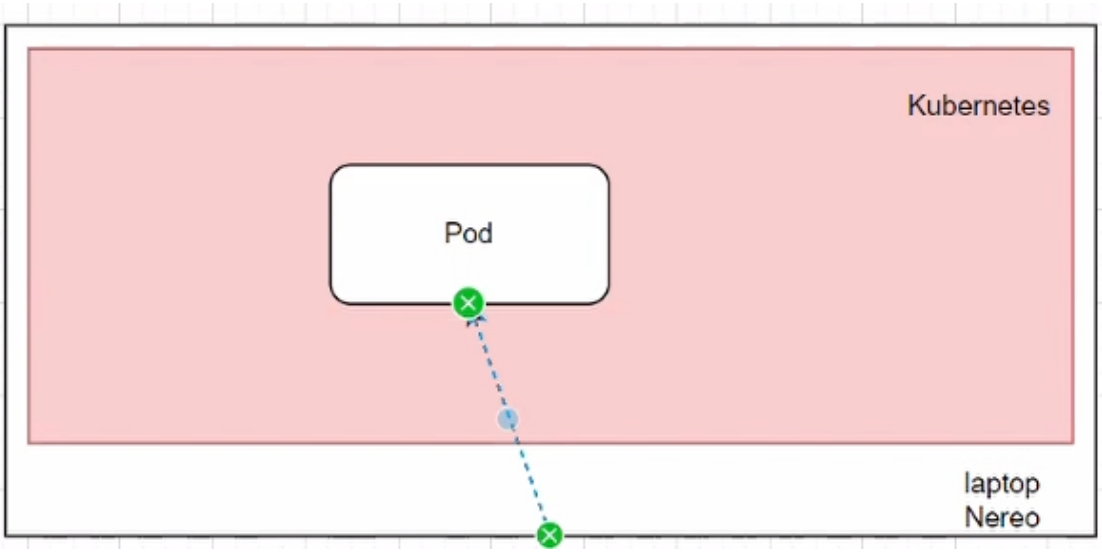
7. Para especificar los recursos que utiliza el POD como la memoria, el vcpu y el gpu, se utiliza el ResourceQuota y el ResourceLimit, especifican la cuota inicial y final respectivamente, si un POD excede las cuotas, muere. Ejemplo:



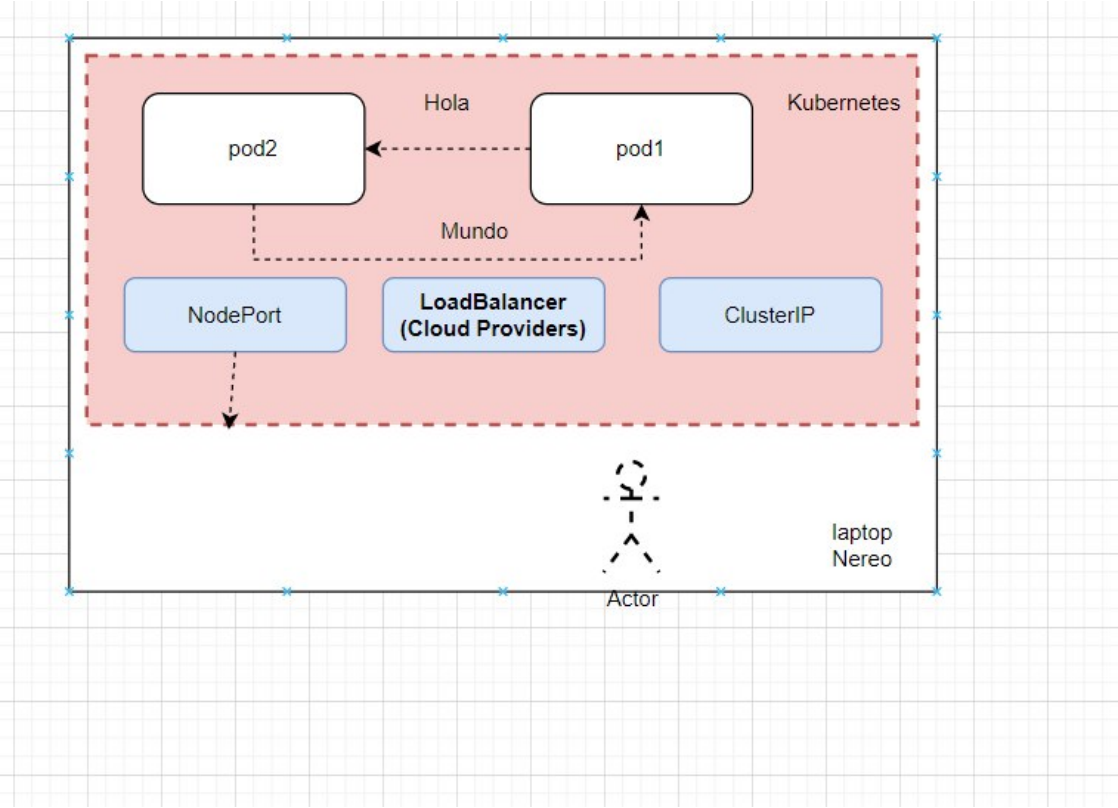
8. Es importante para la experiencia del usuario realizar un ***Load Balancer*** y ***Failover***, qué es repartir el tráfico que tiene un sistema entre el número de réplicas de un POD que tiene un deployment, y que cuando un deployment muere, pasar el tráfico efectivamente a otro. Para esto, se utiliza, como mediador entre el usuario y el deployment un ***service***, este realiza health ckecks y en cada consulta que realiza un usuario, si uno de los POD del deployment esta muerto cuando entra la consulta, el service no lo toma en cuenta.



9. Existen tres de servicios generales dentro de Kubernets para poder tener interacción con los PODs dentro de un deployment, ya que no se puede acceder directamente a un POD. Esto NO se puede:

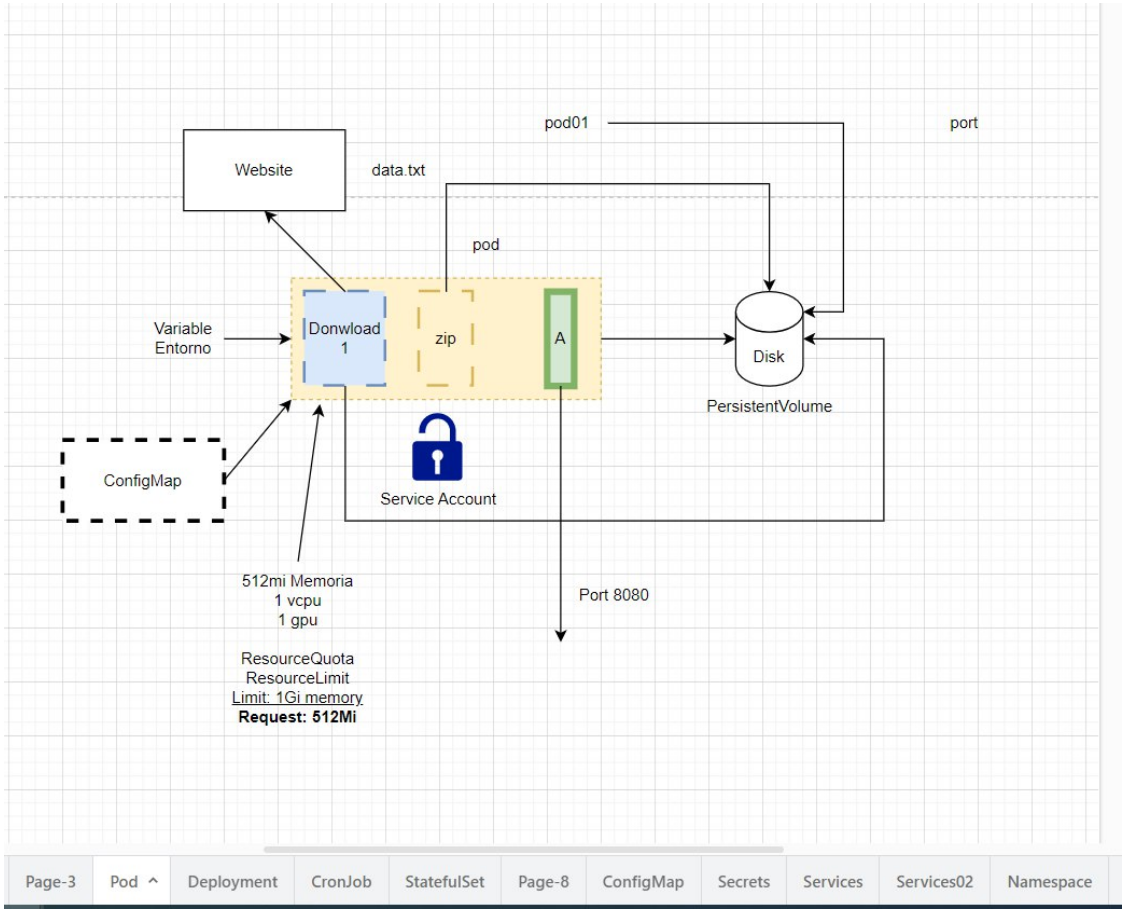


Entonces, para conectar un usuario con lo PODs dentro de un deployment, el primer servicio es un ***NodePort***, es el que utiliza el usuario para poder conectarse a los servicios del deployment. También tenemos el **ClusterIP**, que permite la interacción entre nodos. El último de los tres, es el ***LoadBalancer*** que es un servicio CloudProviders, es decir, un servicio proveído por las nubes para permitir la interacción de entes externos.



10. En Kubernets, se pueden tener diferentes agrupaciones de recursos, estos nos lleva al concepto de ***namespace***, que son espacios dónde se pueden crear todos los objetos de lo que se ha hablado, y son privados de cada namespace. En este ejemplo, ninguno de los namespace, puede ver lo que tiene el otro.

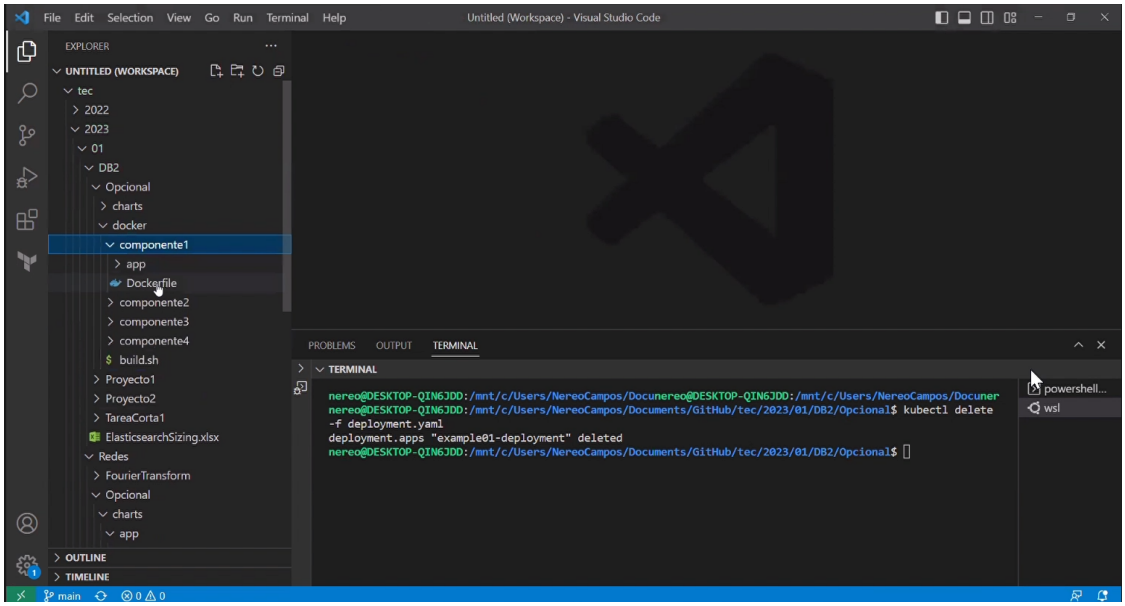
11. A manera qué, agregando y juntando los instrumentos que se han visto, podríamos tener el siguiente diagrama de un sistema:



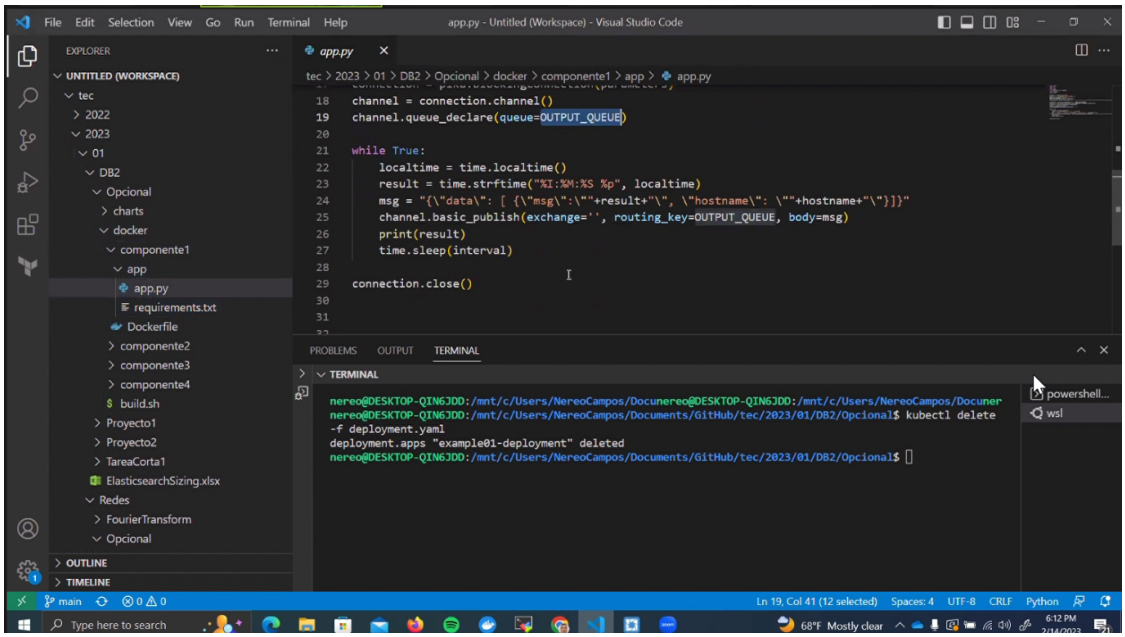
12. El profesor mostró un ejemplo semejante a la base del proyecto opcional. El ejemplo completo fue enviado por Telegram.

    Aquí se explica de manera general el ejemplo, con algunas capturas y diagramas.

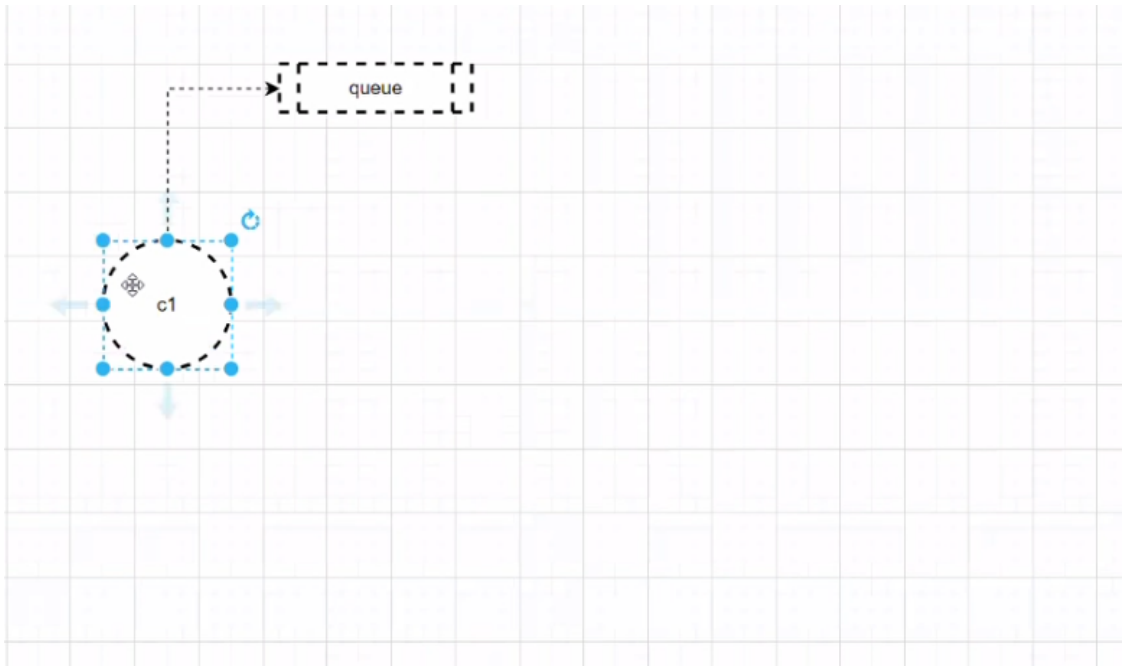
    El proyecto es un docker con cuatro componentes.



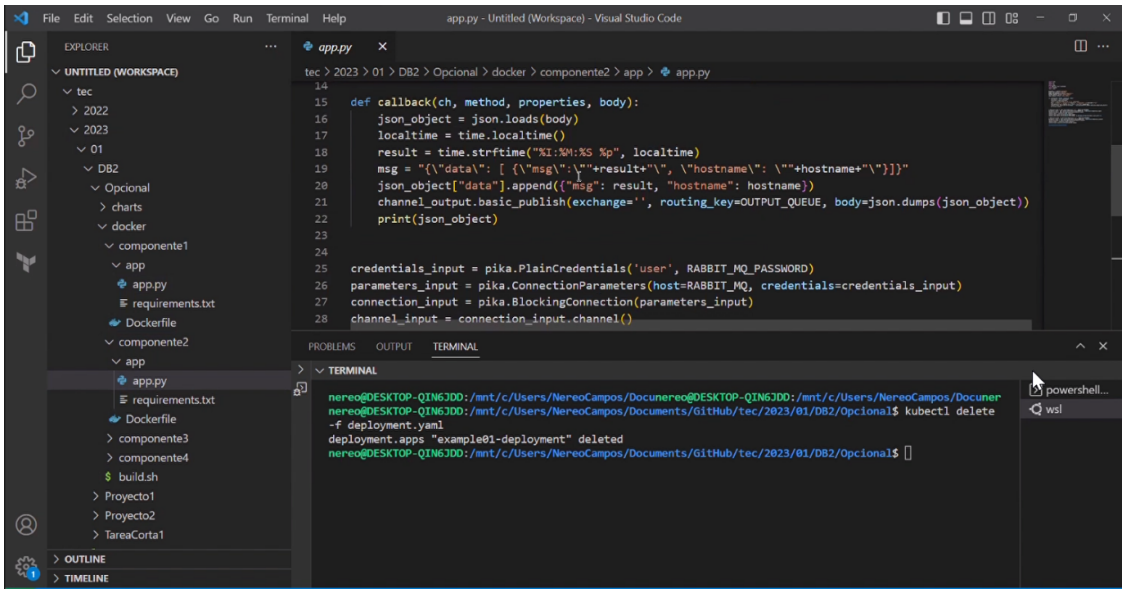
    En el ***primer componente*** se crea una conexión hacia la cola de salida y se genera un mensaje jason para publicar en esa cola de salida.



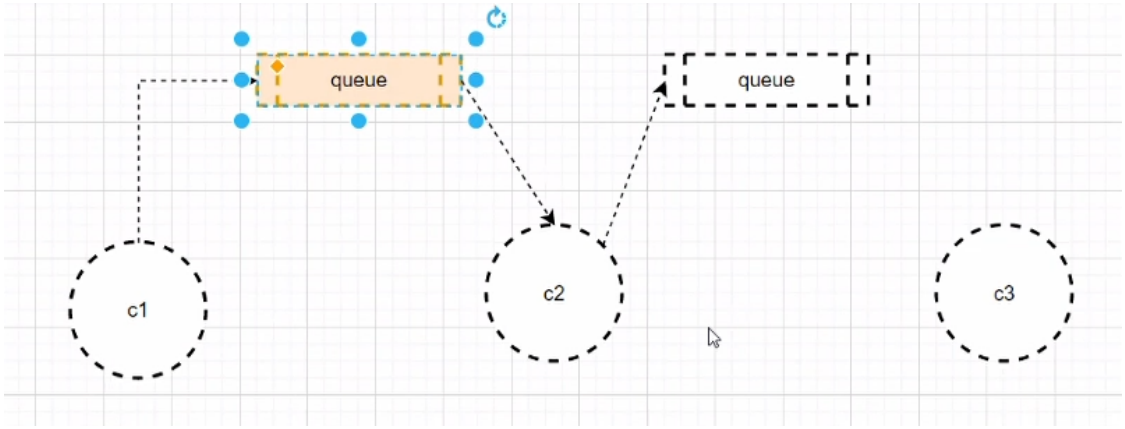
A manera de diagrama, se tiene lo siguiente:



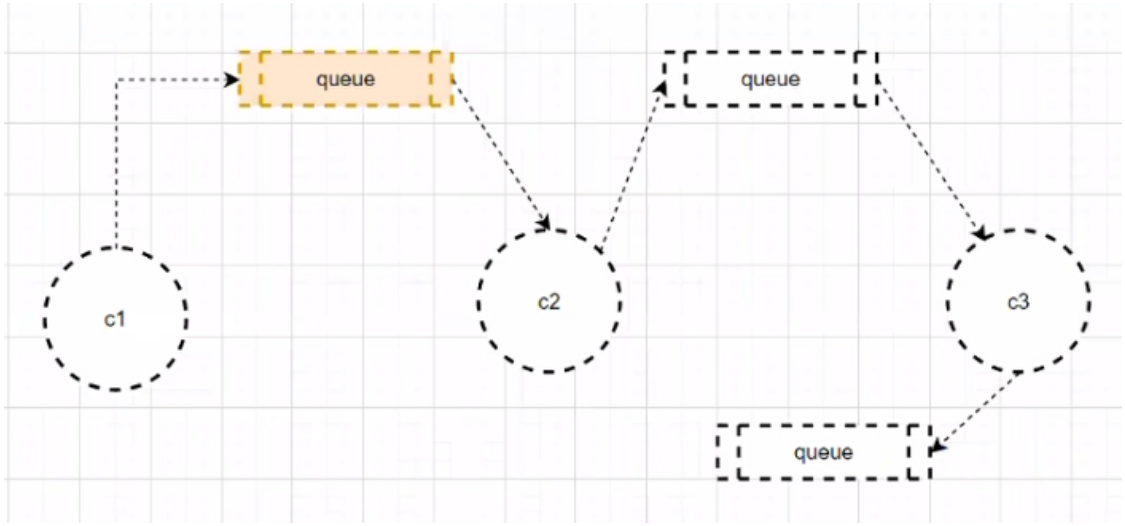
El ***segundo componente***, lo que hace es recibir una cola de entrada (que sería la cola de salida del componente 1) y tener una cola de salida. Una vez se empieza a ejecutar lee la cola de entrada y queda en espera de un mensaje, cuando este llega, se llama a la función callback, que calcula la hora, y se la agrega al mensaje recibido, también le agrega el hostname, y después lo manda a la cola de salida. A continuación la función callback:



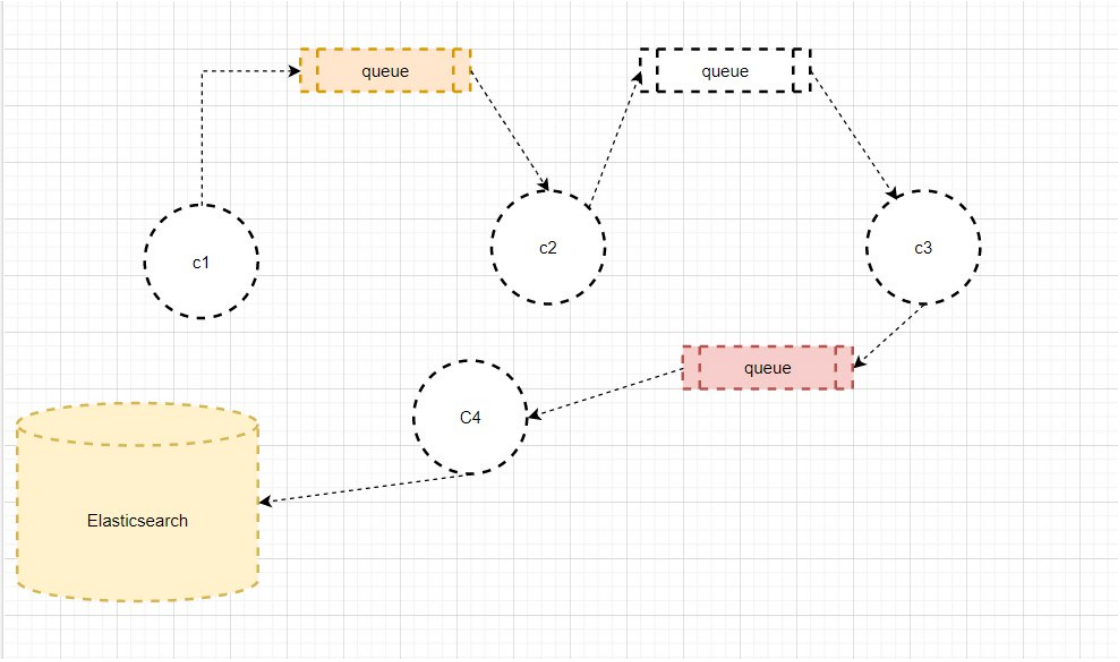
El diagrama entonces, iría así:



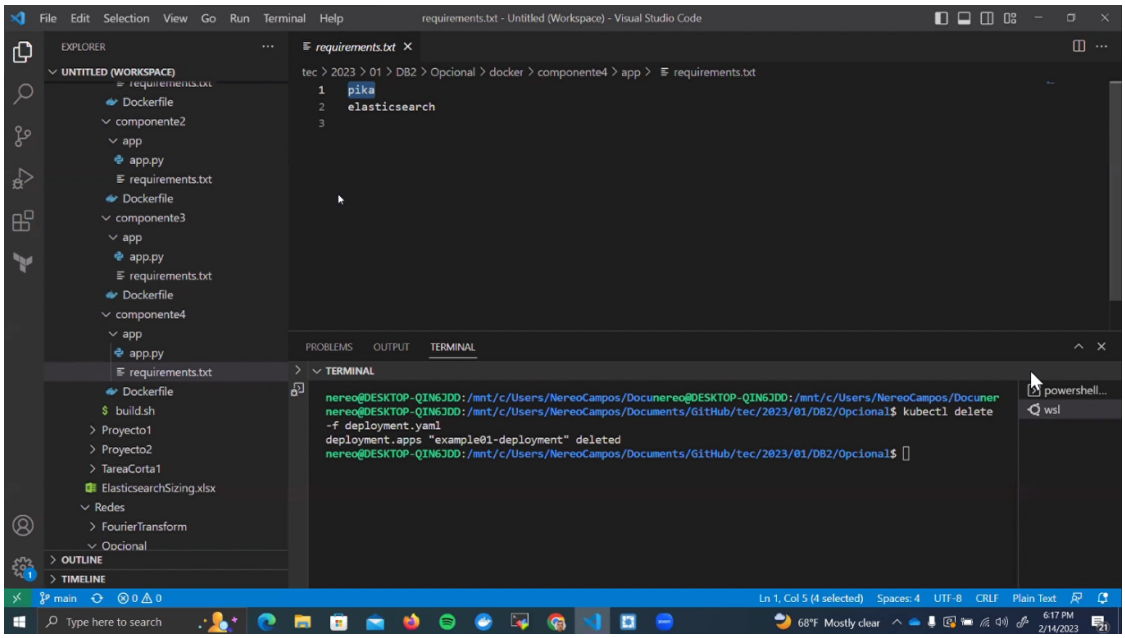
El ***tercer componente*** hace exactamente lo mismo qué el componente 2, y ya tenemos el diagrama así:



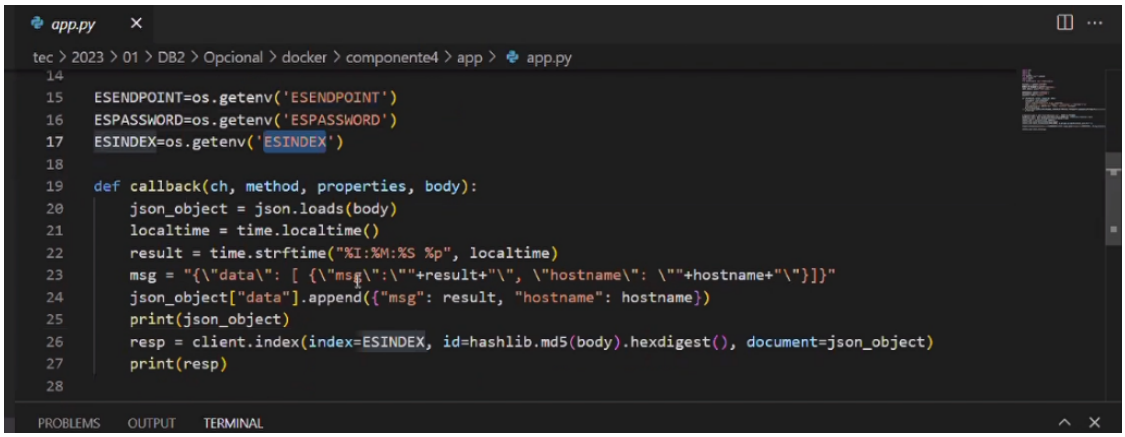
Por último, el ***cuarto componente***, a grandes rasgos lo que hará es conectarse a ***Elasticsearch*** para publicar el mensaje. A manera de diagrama sería así:



En el requirements instala **pika** (qué es la biblioteca para trabajar con colas en python, los otros tres componentes también la tienen) y elasticsearch, que es para conectarse a elasticsearch.



Luego igualmente se conecta a una cola de entrada que es de dónde lee el mensaje. Recibe tres variables de elasticsearch que son las que tienen iniciales ES, que serían el ENDPOINT, el password y el nombre del índice. Y en la función callback lo que hace es guardar el documento en elasticsearch. Observe la imagen:



Por último el profe mostró algunas instalaciones necesarias como por ejemplo la de un ECK en Kubernetes cluster o instalar Kibana, y la manera de correr automáticamente comandos para ejecutar de manera más eficientes esas instalaciones. El archivo completo fue enviado por telegram, allí se puede encontrar toda la configuración que tiene el proyecto opcional para correr.