Premisa:

Las pruebas técnicas pretende es comprender el proceso de razonamiento y la capacidad de resolución de problemas de los candidatos. Por lo tanto, es importante seguir su progreso y elaborar sus pasos analíticos mientras logra la solución. Si cree que algunos de los puntos de esta prueba son demasiado para una prueba, que no tiene el tiempo o las herramientas adecuadas para ofrecer una solución práctica, no dude en detallar cómo llegaría a una solución práctica en el escenario hipotético en el que tiene todo lo necesario para resolver el problema.

Esta prueba costa de 2 partes, la primera de análisis y el cual es aspirante cuenta con 1 hora. La parte 2 corresponde a un pequeño escenario técnico y el aspirante contará con 2 horas para su desarrollo.

******* SYSADMIN *******

Usted está a cargo de un sistema basado en microservicios en JAVA, con conexiones de base de datos en MongoDB, intercambio de mensajes con brokers de Kafka y RabbitMQ, y comunicándose por medio de servicios Restful.

-> ¿Que métricas usted considera críticas para monitorear en el sistema y con qué herramientas lo ejecutaría?, justifique su respuesta.

******* NETWORKING + SYSADMIN + CLOUD *******

Analice, plantee y diseñe una infraestructura que considere lo siguiente (Para efectos de cálculo, use instancias base de cada generación):

Está a cargo en el despliegue de una nueva plataforma que está compuesto de:

- ✓ Un clúster de Kubernetes (Contiene el backend y frontend)
- ✓ Un clúster de Kafka
- ✓ Una base de datos en PostgreSQL (RDS o EC2)
- ✓ Un clúster de MongoDB
- ✓ 4 nodos EC2 que contiene servicios.

Debe cumplir:

- Debe estar en una regios de AWS ubicado en Norteamérica.

- Los costos deben ser optimizados.

- Para los clústeres se debe tenerse 3 nodos cada uno.

- Se dispone de una red 10.100.0.0/19 para K8S, mínimo distribuido en 3 zonas de zonas sin desperdicio de IPs, 3 subredes para segmentos públicos y 3 para segmentos privados.

- Se dispone de una red 10.100.32.0/22 para servicios, se debe crear 4 subredes las cuales contienen (1 productiva, 1 Testting and QA, y 2 para BCP y/o crecimiento). Cada subred debe estar distribuido mínimo en 3 zonas de zonas sin desperdicio de IPs, 3 subredes para segmentos públicos y 3 para segmentos privados.

- Las Instancias no deben ser tipo ráfaga.

* Como plus y si es posible, entregar un estimado de la solución.

******* DOCKER + TROUBLESHOOTING *******

Se está corriendo una aplicación en Python Flask (El código es irrelevante para este ejercicio, puede ser cualquiera) usando **gunicorn** y **haproxy** todo dentro de un mismo contenedor. El **dockerfile** está como se muestra:

Archivo: **Dockerfile**

```
FROM alpine
RUN apk add py3-pip build-base python3-dev libffi-dev openssl-dev
RUN apk add nginx
RUN mkdir -p /opt/api
WORKDIR /opt/api
ADD api/requirements.txt /opt/api
RUN pip3 install --no-cache-dir -r requirements.txt
ADD api/. /opt/api
ADD ./docker-entrypoint.sh /bin/docker-entrypoint
ADD ./haproxy.conf /etc/haproxy/haproxy.cfg
EXPOSE 80
CMD ["/bin/docker-entrypoint"]
```

Archivo: **docker-entrypoint.sh**

```
#!/bin/sh

echo"Starting gunicorn..."
gunicorn -w 5 -b 127.0.0.1:9000 app:app --daemon
sleep 3

echo"Starting haproxy..."
haproxy -f "/etc/haproxy/haproxy.cfg" &
```

Archivo: **haproxy.cfg**

```
global
   maxconn 8192

defaults
   log stdout format raw local0
   mode    http
   option  httplog
   option  forwardfor
   option  httpclose
   option  dontlognull

   timeout connect 10s
   timeout client  150s
   timeout server  150s

   errorfile 400 /usr/local/etc/haproxy/errors/400.http
   errorfile 403 /usr/local/etc/haproxy/errors/403.http
   errorfile 408 /usr/local/etc/haproxy/errors/408.http
   errorfile 500 /usr/local/etc/haproxy/errors/500.http
   errorfile 502 /usr/local/etc/haproxy/errors/502.http
   errorfile 503 /usr/local/etc/haproxy/errors/503.http
   errorfile 504 /usr/local/etc/haproxy/errors/504.http
```

```
   maxconn 8192

frontend app-http
   bind *:80

   acl is_app  path_beg -i  /
   use_backend flask_backend  if is_app


backend flask_backend
   server docker-app cll-dsb-auth:9000 check verify none
```

La aplicación funciona correctamente, sin embargo, algunos momentos esta inestable o el servicio deja de responder retornando errores 500, el contenedor sigue corriendo, aunque el servicio no esté operativo.

Con lo anterior:

- Hacer un análisis con expuesto que errores considera que se están cometiendo.
- Que mejoras y/o optimizaciones se pueden llevar a cabo.
- Como SysAdmin, ¿Cómo se manejaría los logs y de qué forma se presentará para ser usado en una herramienta de monitoreo?

******* KUBERNETES ***** (Practico – 2 horas)**

Previamente el aspirante debe tener y validar.

- ✓ Un usuario GitHub.
- ✓ Un usuario de DockerHub.
- ✓ Kubernetes local (minikube) para ejecutar validar lo entregado.
- ✓ Haber instalado en su equipo kubectl y helm.

Con el punto anterior (`DOCKER + TROUBLESHOOTING`) desarrolle.

- Aplique los mejoras y/o correcciones necesarias para desplegar este servicio usando docker. Para el ejemplo en flask puede usar un simple "`hello world`" que encuentre en internet.
- Con el o los contenedores creados en el paso previo, genere los manifiestos de Kubernetes necesarios para desplegar este servicio.
- Incluya los *probes* de kubernetes necesarios para validar que el servicio este sirviendo correctamente.
- Usando el siguiente enlace https://www.magalix.com/blog/monitoring-of-kubernetes-cluster-through-prometheus-and-grafana y usando *minikube*, despliegue el prometheus y visualice las matricas del servicio creado.

Una vez el aspirante considere que ha finalizado con la tarea, debe subir a “DockerHub” los contenedores creados en un proyecto público para que el calificador los pueda descargar y evaluar. De igual manera todos los archivos fuente generados deben ser colocados en el “GitHub” del aspirante en un proyecto público para su verificación. Los manifiestos de Kubernetes que despliegue Pods en sus diferentes tipos de objetos, deben apuntar a la dirección de “DockerHub” del primer paso.