Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Sistemas Operativos 1
Catedrático:
Auxiliar:
Primer Semestre 2021

Proyecto 2
Manual de Tecnico

GRUPO 13

201313692	Jossie Bismarck Castrillo Fajardo
201314556	Carlos Gabriel Peralta Cambrán
201314697	Katherine Mishelle Serrano del Cid
201020331	Cristian Alexander Azurdia Ajú

CREACIÓN DE CLUSTER

- **Creación de cluster en Gcloud:**

  gcloud container clusters create proyectosopes1 --num-nodes=2 --tags=allin,allout --enable-legacy-authorization --enable-basic-auth --issue-client-certificate --machine-type=n1-standard-2 --no-enable-network-policy

- **Obtener las credenciales del cluster:**

  gcloud container clusters get-credentials tesisdemo --zone us-east1-b --project grupo28-ing-usac-so2

INSTALACIÓN DE LINKERD

- **Instalar CLI de Linkerd**

  curl -sL run.linkerd.io/install | sh

- **Validar Instalación**

  linkerd versión

- **Validar si el cluster cumple con lo necesario para Linkerd**

  linkerd check –pre

- **Instalar control plane dentro del cluster**

  linkerd install | kubectl apply -f -

- **Agregar la extension para el uso del dashboard**

  linkerd viz install | kubectl apply -f –

- **Activar dashboard**

  linkerd viz dashboard &

INSTALACIÓN DE CHAOS MESH

- **Instalación**

  curl -sSL https://mirrors.chaos-mesh.org/v1.1.2/install.sh | bash

- **Verificar instalación**

  kubectl get pod -n chaos-testing

- **Activar dashboard**

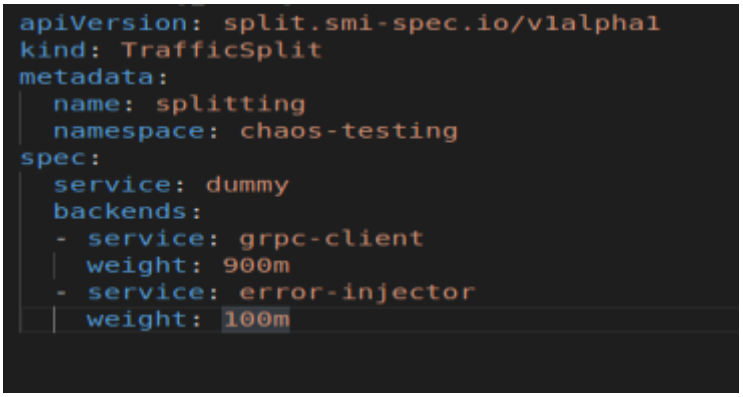
  kubectl port-forward -n chaos-testing svc/chaos-dashboard 2333:2333

DEPLOY DE ARQUITECTURA

kubectl apply -f deploy.yaml

TRAFFIC SPLITTING

De momento se divide la carga entre la cantidad activa de canales de servicio.



INGRESS

- **Instalar Helm**

  https://helm.sh/docs/intro/quickstart/

  wget https://get.helm.sh/helm-v3.2.4-linux-amd64.tar.gz

  tar -xzvf helm…..

  sudo mv linux-amd64/helm /sbin

- **Crear namespace para Nginex-Ingress**

  kubectl create ns nginx-ingress

- **Actualizar repositorios de Helm**

  helm repo update

- **Instalar Ingress de Nginx**

  helm install nginx-ingress ingress-nginx/ingress-nginx -n nginx-ingress

- **Inyeccion del Ingress con Linkerd**

  kubectl get deployment nginx-ingress-ingress-nginx-controller -n nginx-ingress -o yaml | linkerd inject --ingress - | kubectl apply -f -

Respuestas

- Cómo funcionan las golden metrics, cómo pueden interpretarse las cuatro pruebas de faulty traffic utilizando de base las gráficas y métricas que muestra Linkerd y Grafana en los dashboards.

  Las golden metrics toman información sobre el porcentaje atendido con éxito, la tasa promedio de peticiones por segundo y la latencia para los percentiles 50, 95 y 99 siendo estas 3 la tasa media de solicitudes atendidas para el 50%, 95% y 99%.

- Mencionar al menos tres patrones de conducta que fueron descubiertos.

  1. Con todos los experimentos se alteraba la latencia para los 3 percentiles.
  2. Durante el tiempo de ejecucion de los experimentos disminuye la cantidad de peticiones por segundo.
  3. Los pods se recargaban en Linkerd al momento de la ejecución de los experimentos para estabilizarse al final.

TERCERA PARTE

- ¿Qué sistema de mensajería es más rápido?

  El más rápido sería Redis pub/sub con la capacidad de manejar hasta 81 mil sets por segundo y 110 mil gets por segundo.

- ¿Cuántos recursos utiliza cada sistema?

- ¿Cuáles son las ventajas y desventajas de cada sistema?
  gRPC
  - Ventajas
    - Transmisión bidireccional
    - Utiliza HTTP/2
    - Permite integración de tecnologías de seguridad
    - Alto rendimiento
    - Puede ejecutarse en cualquier entorno
  - Desventajas
    - Dependencia de una red estable y potente
    - No compatible con multicasting

  Kafka
  - Ventajas
    - Se maneja de forma distribuida
    - Tolerante a fallas
    - Sistema de mensajería
    - Adsorción de picos de carga en un sistema
  - Desventajas
    - No soporta mensajes de mas de 1MB
    - Al ejecutar operaciones de rebalanceo, se pierde performance
    - Pocas opciones de monitoreo

  Redis pub/sub
  - Ventajas
    - Multiples funcionalidades (memcache, queue, pubsub, etc)
    - Maneja cualquier número de suscriptores a un canal.
    - Sus mensajes se manejan por medio de "fire-and-forget"
  - Desventajas
    - La persistencia puede afectar el rendimiento
    - Configuración de los clusters es compleja
    - Falta de opciones de seguridad avanzadas y control de accesos

- ¿Cuál es el mejor sistema?

  Como grupo consideramos que no podemos indicar cual seria el mejor, ya que cada uno posee una finalidad de uso y aplicación distinta, son considerados mas óptimos según el uso que se le vaya a dar.

CUARTA PARTE

- ¿Cuál de las dos bases de datos se desempeña mejor y por qué?

  En tema de desempeño redis supera a mongodb, redis es superior en cuestión de lecturas para todo tipo de cargas de trabajo, y mejor para las escrituras conforme la carga de trabajo va incrementando. A pesar de que mongo usa todos los núcleos del sistema cuando tiene suficiente capacidad puede superar a redis en escrituras. Finalmente redis sería superior dado que logra más ejecutándose sobre un solo núcleo sin saturarlos, aunque utiliza una cantidad de memoria RAM mayor a mongo

para cantidad igual de almacenamiento.

CHAOS ENGINEERING

- ¿Cómo cada experimento se refleja en el gráfico de Linkerd, qué sucede?

  Las métricas de funcionamiento se alteran, ya sea en la tasa de éxitos, peticiones por segundo y la latencia para los 3 percentiles.

- ¿Cómo cada experimento es distinto?

  Principalmente por el objetivo al cual están orientados, faulty traffic busca generar peticiones erróneas por lo cual afectará la tasa de peticiones exitosas atendidas sin afectar directamente al resto de la arquitectura.
  Pod Kill busca matar pods mientras que Pod Failure genera fallas aleatorias internas a los pods, estos dos experimentos afectan únicamente a los pods pero a nivel de sistema pueden afectar el tiempo de respuesta dado que la carga recae sobre los pods activos. Network emulation se enfoca en generar fallas a nivel de red, ya sea como latencia, jitter, pérdida de paquetes, etc. afectando directamente la red sobre la cual se comunican los servicios internos. DNS Chaos afecta únicamente a la traducción de direcciones afectando la IP o dirección hacia la cual se envía un paquete.

- ¿Cuál de todos los experimentos es el más dañino?

  El más dañino es Pod kill dado que elimina completamente un pod, quitando un canal de comunicación hacia las bases de datos, los demás experimentos solo bajan levemente el rendimiento.