TIC

TALENTO TECH

Misión 1

Arquitectura en la nube

Innovador

Tema 2: Administración de aplicaciones con Kubernetes

UTraining Entrenamiento en Innovación y Tecnología | UNIVERSIDAD DE ANTIOQUIA | Universidad de Caldas | ubicua Technology

Campista, llegó el momento de retar tus conocimientos y que los pongas aprueba a través de los diferentes recursos que encontraras en este espacio como son: conceptos, ejemplos, herramientas, actividades prácticas y retos, los cuales te ayudaran alcanzar los objetivos trazados en el nivel innovador.

Arquitecturas escalables: autoescalado, balanceo de carga, patrones de partición

Las arquitecturas escalables son fundamentales en el desarrollo de aplicaciones modernas que requieren adaptarse eficientemente a cambios en la demanda y en la carga de trabajo. El autoescalado permite ajustar automáticamente los recursos del sistema en función de la necesidad, garantizando un rendimiento óptimo sin intervención manual. El balanceo de carga distribuye el tráfico entre múltiples servidores, asegurando la disponibilidad y evitando sobrecargas.

Los patrones de partición, por su parte, dividen los datos y las tareas entre distintos nodos para mejorar la eficiencia y reducir los tiempos de respuesta. Juntas, estas estrategias permiten construir sistemas robustos, flexibles y capaces de manejar un alto volumen de tráfico y datos de manera eficiente.

CONCEPTOS CLAVE

Autoescalado: Ajuste automático de recursos.

Balanceo de Carga: Distribución de tráfico.

Patrones de Partición: División de datos para eficiencia.

AUTOESCALADO EN KUBERNETES

Exploraremos cómo funciona el autoescalado en Kubernetes, incluyendo Horizontal Pod Autoscaler (HPA) y Vertical Pod Autoscaler (VPA).

HORIZONTAL POD AUTOSCALER (HPA)

HPA ajusta el número de pods en función de la carga de CPU o métricas personalizadas. Es esencial para manejar cargas variables.

VERTICAL POD AUTOSCALER (VPA)

VPA ajusta los recursos asignados a los pods (CPU y memoria) para optimizar el rendimiento y la eficiencia.

CASO DE ESTUDIO: HPA Y VPA

Analizaremos un caso de estudio donde una aplicación utiliza HPA y VPA para escalar automáticamente según la carga.

BALANCEO DE CARGA EN KUBERNETES

Aprenderás a configurar y utilizar balanceadores de carga en Kubernetes para distribuir el tráfico de manera eficiente.

TIPOS DE BALANCEADORES DE CARGA

- ClusterIP: Acceso interno.
- NodePort: Acceso externo.
- LoadBalancer: Balanceo de carga externo.

TALLER PRÁCTICO

Se te introducirá a los patrones de partición y cómo aplicarlos en Kubernetes para mejorar la eficiencia.

PATRONES DE PARTICIÓN EN KUBERNETES

Se te introducirá a los patrones de partición y cómo aplicarlos en Kubernetes para mejorar la eficiencia.

EJEMPLOS DE PATRONES DE PARTICIÓN

- Sharding: División de datos.
- Hash Partitioning: Distribución basada en hash.
- Range Partitioning: División por rangos.

EVALUACIÓN Y REFLEXIÓN FINAL

Completarás un cuestionario de evaluación y participarás en una reflexión en grupo sobre la aplicación práctica de las técnicas aprendidas.

Arquitecturas Resilientes - Implementación de Tolerancia a Fallos y Recuperación ante Desastres

Las arquitecturas resilientes son esenciales para garantizar la continuidad y estabilidad de los sistemas en entornos tecnológicos críticos, donde las interrupciones pueden tener un impacto significativo. La implementación de tolerancia a fallos permite que los sistemas sigan funcionando correctamente, incluso cuando se producen errores o fallos en componentes individuales, mediante técnicas como la redundancia y la replicación.

Por otro lado, la recuperación ante desastres se centra en la preparación y los procedimientos necesarios para restaurar las operaciones después de un incidente mayor, como un ciberataque o un fallo masivo de infraestructura. Combinando estas estrategias, las arquitecturas resilientes aseguran que las aplicaciones y servicios sean robustos, minimizando el tiempo de inactividad y garantizando una alta disponibilidad y continuidad operativa.

INTRODUCCIÓN A LA TOLERANCIA A FALLOS Y RECUPERACIÓN ANTE DESASTRES

La resiliencia es crucial en arquitecturas contenerizadas. Exploraremos cómo asegurar la continuidad del servicio y la rápida recuperación ante fallos.

IMPORTANCIA DE LA RESILIENCIA

La resiliencia asegura que las aplicaciones continúen funcionando ante fallos y desastres, minimizando el impacto en los usuarios y el negocio.

CONCEPTOS CLAVE

- Tolerancia a Fallos: Capacidad de un sistema para seguir operando ante fallos.
- Recuperación ante Desastres: Estrategias para restaurar servicios tras un fallo grave.

CASOS DE ESTUDIO

Analizaremos ejemplos prácticos donde se aplicaron estrategias de tolerancia a fallos y recuperación ante desastres en Kubernetes.

ESTRATEGIAS DE TOLERANCIA A FALLOS EN KUBERNETES

Exploraremos el uso de Horizontal Pod Autoscaler (HPA), Probes y Topology Constraints para implementar tolerancia a fallos.

HORIZONTAL POD AUTOSCALER (HPA)

HPA ajusta automáticamente el número de pods en función de la carga. Configuración YAML:

yaml

Copiar código

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: example-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: example-deployment
  minReplicas: 1
```

PROBES

Probes verifican la salud de los contenedores. Tipos: Liveness, Readiness y Startup. Configuración YAML:

yaml

Copiar código

```
livenessProbe:
  httpGet:
    path: /healthz
    port: 8080
  initialDelaySeconds: 3
  periodSeconds: 3
```

TOPOLOGY CONSTRAINTS

Topology Constraints aseguran que los pods se distribuyan adecuadamente en el clúster.

Configuración YAML:

yaml

Copiar código

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: "kubernetes.io/hostname"
  whenUnsatisfiable: DoNotSchedule
  labelSelector:
    matchLabels:
      app: example
```

ESTRATEGIAS DE RECUPERACIÓN ANTE DESASTRES EN KUBERNETES

Topology Constraints aseguran que los pods se distribuyan adecuadamente en el clúster.

Configuración YAML:

yaml

Copiar código

```
topologySpreadConstraints:
- maxSkew: 1
  topologyKey: "kubernetes.io/hostname"
  whenUnsatisfiable: DoNotSchedule
  labelSelector:
    matchLabels:
      app: example
```

CONFIGURACIÓN DE BACKUPS

Los backups son esenciales para la recuperación ante desastres. Ejemplo de configuración YAML:

yaml

Copiar código

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: backup-job
spec:
  schedule: "0 2 * * *"
  jobTemplate:
    spec:
```

```
template:
 spec:
  containers:
  - name: backup
    image: backup-image
    args:
    - /bin/sh
    - -c
    - "backup-command"
  restartPolicy: OnFailure
```

EVALUACIÓN Y REFLEXIÓN FINAL

Realizaremos una evaluación rápida y reflexionaremos sobre la importancia de la resiliencia en arquitecturas contenerizadas.

SUMMARY

Hemos explorado la importancia de la resiliencia, estrategias de tolerancia a fallos y recuperación ante desastres en Kubernetes. La implementación de estas prácticas asegura la continuidad del servicio y la rápida recuperación ante fallos.

Aprendizajes prácticos

Práctica 1: Escalado en K8s

Despliegue de aplicaciones con autoescalado y balanceo de carga

Objetivo:

Guía para realizar diferentes pruebas de concepto utilizando Kubernetes con la aplicación creada inicialmente. Las pruebas incluyen HPA (Horizontal Pod Autoscaler), reinicio de pods, y escalados a demanda.

https://articulateusercontent.com/rise/courses/fuq_GhcPb-nvjhVwubAVDtFITDdwwLOH/YLtefufJ3WwecpSz-Aprendizaje%2520pr%25C3%25A1ctico.%2520Escalado_Kubernetes.pdf

Práctica 2: Probes Kubernetes

Despliegue de aplicaciones en EKS, estados de los contenedores, pruebas de vida de contenedores

https://articulateusercontent.com/rise/courses/fuq_GhcPb-nvjhVwubAVDtFITDdwwLOH/LQzACO2RVx7Voyf1-Aprendizaje%2520pr%25C3%25A1ctico.%2520Probes_Kubernetes.pdf

Práctica 3: Práctica Istio

Practica sobre exposición de servicios en EKS – Istio

https://articulateusercontent.com/rise/courses/fuq_GhcPb-nvjhVwubAVDtFITDdwwLOH/NYeCxkPJK7VJM3ib-Aprendizaje%2520pr%25C3%25A1ctico.%2520Practica_Istio.pdf