Kubernetes Automation Script with KEDA and Kafka Integration

Overview

This script automates Kubernetes operations by automating application deployment, service creation, and Kafka-based autoscaling with KEDA, simplifying resource management efficiently.

Features

1. Prerequisite Check:
   - Ensures essential tools (`kubectl` and `helm`) are installed.
2. KEDA Installation:
   - Deploys KEDA into the Kubernetes cluster for event-driven autoscaling.
3. Custom Deployments:
   - Enables the creation of Kubernetes Deployments with configurable Docker images, resource limits, and exposed ports.
4. Service Management:
   - Creates internal-only services using the ClusterIP type.
5. Kafka-based Autoscaling:
   - Integrates with Kafka to enable autoscaling of pods based on message lag.
6. Health Monitoring:
   - Verifies the health of deployments and associated pods.

Script Commands and Usage

1. **Install Tools**:
   - Command: `./script.sh install`
   - Validates the installation of `kubectl` and `helm`. Outputs errors if any tools are missing.
2. **Setup KEDA**:
   - Command: `./script.sh setup`
   - Installs KEDA in the cluster using Helm, placing it under the `keda` namespace.
3. **Create Deployment**:
   - Command: `./script.sh deploy <namespace> <deployment_name> <image> <tag> <port> <cpu_request> <memory_request> <cpu_limit> <memory_limit> <bootstrap_servers> <topic> <consumer_group> <lag_threshold>`
   - Allows the deployment of applications with Kafka-driven autoscaling. The parameters include:
     - `namespace`: The namespace for the deployment (e.g., `busybox`).
     - `deployment_name`: The name of the deployment (e.g., `busybox`).
     - `image` and `tag`: Docker image and tag (e.g., `busybox:latest`).
     - `port`: The port exposed by the container.
     - CPU and memory requests/limits to allocate resources.
     - Kafka-related parameters like bootstrap servers, topic and consumer group.

4. **Check Deployment Health**:
    - Command: `./script.sh health <namespace> <deployment_name>`
    - Monitors the health of the specified deployment and ensures pods are running correctly.

Deployment Details

- **Kubernetes Deployment**:
    - Allows custom configurations for DockerHub images, CPU/memory requests, and exposed ports.
- **Service**:
    - Creates a ClusterIP service to enable internal communication within the Kubernetes network.
- **KEDA ScaledObject**:
    - Configures Kafka as the fixed event source for autoscaling. Includes parameters such as:
        - `bootstrapServers`: Kafka brokers (e.g., `kafka.svc:9092`).
        - `topic`: The Kafka topic monitored for scaling triggers.
        - `consumerGroup`: The Kafka consumer group.
    - Autoscaling limits:
        - Minimum replicas: 1
        - Maximum replicas: 10

Example Commands

1. **Install Prerequisites**:
    - `./script.sh install`
2. **Setup KEDA**:
    - `./script.sh setup`
3. **Create a Busybox Deployment**:
    - `./script.sh deploy busybox busybox busybox latest 8080 50m 32Mi 100m 64Mi kafka.svc:9092 my-test-topic my-test-cg 10`
4. **Check Deployment Health**:
    - `./script.sh health busybox busybox`

Workflow Overview

1. Install tools: Ensures `kubectl` and `helm` are installed on the system.
2. Setup KEDA: Deploys KEDA to enable event-driven scaling.
3. Create Deployment:
    - Generates YAML files for deployments, services, and autoscaling configurations.
    - Applies these configurations using `kubectl`.
4. Configure Autoscaling:
    - Monitors the Kafka topic for message lag and scales replicas dynamically.
5. Health Monitoring:
    - Checks the status of deployments and pods, ensuring the environment is running as expected.

Requirements

1. A Kubernetes cluster with `kubectl` configured.
2. Helm installed for Kubernetes package management.
3. A Kafka service available within the Kubernetes cluster.
4. A metrics-server deployment installed

Known Limitations

1. Services created are restricted to internal access (ClusterIP only).
2. Kafka is the only event source used for autoscaling in this script.