Documentation for Drkiq Project

This project was following the Dockerizing a Ruby on Rails Application by Nick Janetakis based on the GitHub repo by TomFern.

Perquisites

The blog is explaining how to spin up a dockerized Rails application with nginx container acts as a reverse proxy in front of it. Also, a redis and postgres containers for caching and storage.

Following the tutorial, you will have your own docerized version of the rails application and nginx server pushed to you Dockerhub account, ex: **nohierhassan/dockerizing-ruby-drkiq** and **nohier/dockerizing-ruby-nginx**

Implementation

Infrastructure

Instead of running the application with docker compose as the tutorial, we are going to run it in a Kubernetes environment. We have used **kubeadm** to setup our Kubernetes cluster on **AWS** instances.

The cluster consists of 3 nodes, 1 master and 2 worker nodes, with **flannel** plugin configured as the networking solution, connectivity is established between all nodes in the cluster.

Components

- *Deployments:*

  We have all the pods in deployment objects to ensure that the desired state of the pods is always achieved.

  - drkiq-deployment:

    The main rails application based on **nohierhassan/dockerizing-ruby-drkiq** image

  - sidekiq-deployment:

    Job scheduler application that is also based on **nohierhassan/dockerizing-ruby-drkiq**

  - postgres-deployment:

    The database based on **postgres:12.1** image which **drkiq-deployment** is using

  - redis-deployment:

    The caching database that is used by **drkiq-deployment** and **sidekiq-deployment**

  - nginx-deployment:

    The customized ngingx pod that is used as the reverse proxy by **drkiq-deployment**

- *ConfigMaps:*

We have used ConfigMaps to pass environment variables to containers running inside the pods.

- Postgres-configmap:

  Used by **postgres** container to pass the **POSTGRES_USER** environment variable

- Shared-configmap:

  Used by both **drkiq** and **sidekiq** to pass some application configuration variables

- *Secrets:*

We have used Secrets to pass some secret variables like passwords to the running containers inside the pods.

- Postgres-secret:

Used by **postgres** container to pass the **POSTGRES_PASSWORD** environment variable

- Shared-secret:

Used by both **drkiq** and **sidekiq** to pass some application secret variables variables

- *services:*

We have used services to communicate with the pods both internally and externally.

- Drkiq:

A cluster IP service that routes the internal traffic to port 8010 to **drkiq-container** on port 8010.

- Postgres:

A cluster IP service that routes the internal traffic to port 5432 to **postgres-containe**r o n port 5432.

- redis:

A cluster IP service that routes the internal traffic to port 6479 to **redis-container** on port 6379.

- nginx:

A NodePort service that routes the external traffic sent to any of the cluster nodes on port 30000 port 8020 to **nginx-container**, which routes the traffic to **drkiq-container** on port 8010

- *Persistent Volumes:*

We have used persistent volumes to create some local storage to be used by containers as normal volumes.

  - postgres-pv
    A volume that is supposed to be used by **postgres-container** for persistent storage

  - redis-pv
    A volume that is supposed to be used by **redis-container** for persistent storage

- *Persistent Volume Claims:*

We have used persistent volume claims to bound the local volumes to the containers.

  - postgres-pvc
    A claim that requests the storage for **postgres-container**

  - redis-pvc

    A claim that requests the storage for **redis-container**