**Final Project Sanbercode**

**Kategori Devops**

**Skenario pengerjaan**

1. Lakukan instalasi docker dan docker compose, kemudian buat image container yang berisikan nginx yang support php 7.2 dan MySQL.
2. Kemudian masukkan project Laravel ke dalam docker.
3. Deploy project Laravel yang berbentuk docker tersebut ke dalam Kubernetes.

**Spesifikasi Sistem**

|  |
| --- |
| master.kube 10.8.0.2/16  node1.kube 192.168.226.143/24 |

**Pembahasan**

1. **Instalasi Docker & Docker compose**

Proses instalasi package Docker & Docker compose dapat dilakukan menggunakan perintah sebagai berikut

|  |
| --- |
| $ curl -fsSL https://get.docker.com -o get-docker.sh  $ sudo sh get-docker.sh  $ sudo curl -L "https://github.com/docker/compose/releases/download/1.26.1/docker-compose-$(uname -s)-$(uname -m)" -o /bin/docker-compose |

Kemudian dilakukan pengecekan apakah keduanya telah terinstal dalam path yang sesuai dan jalankan layanan docker sebagaimana berikut ini

|  |
| --- |
| $ docker version | head  $ docker-compose version | head  $ sudo systemctl start docker |
| Client: Docker Engine - Community  Version: 19.03.5  API version: 1.40  Go version: go1.12.12  Git commit: 633a0ea838  docker-compose version 1.25.5, build unknown  docker-py version: 4.2.0  CPython version: 3.7.5  OpenSSL version: OpenSSL 1.1.1 11 Sep 2018 |

**Instalasi Nginx, PHP-FPM, MySQL pada Docker Image**

Pada tahap ini, akan dilakukan skenario deployment dengan arsitektur *single-application* yang memuat keseluruhan dari Nginx, PHP-FPM, dan MySQL. Untuk membuatnya, terlebih dahulu definisikan sebuah file bernama **Dockerfile**

|  |
| --- |
| $ vim Dockerfile |
| FROM php:7.2-fpm-alpine  RUN apk add --update mysql mysql-client nginx  RUN docker-php-ext-install mysqli  RUN docker-php-ext-install -j$(nproc) pdo\_mysql |

Dapat terlihat di sini bahwa terdapat serangkaian proses yang dilakukan, di antaranya

1. FROM php:7.2-fpm-alpine, dimana akan diunduh Docker Image dengan package **php-fpm** di dalamnya
2. RUN apk add --update mysql mysql-client nginx, dimana akan dilakukan instalasi package **mysql server-client** beserta **nginx** dari repository alpine
3. RUN apk add --update mysql mysql-client nginx, dimana akan dilakukan instalasi package **mysql server-client** beserta **nginx** dari repository alpine
4. RUN docker-php-ext-install mysqli, dimana akan dilakukan instalasi php-mysqli module yang berfungsi untuk berkomunikasi dengan database MySQL
5. RUN docker-php-ext-install -j$(nproc) pdo\_mysql, dimana akan dilakukan instalasi php-pdo-mysql module yang berfungsi untuk berkomunikasi dengan database MySQL berbasis PDO

Setelah mendefinisikan **Dockerfile**, akan dibuat docker image dengan perintah

|  |
| --- |
| $ docker build -t sanbercode-project-alpine . |

Hasilnya, sebuah docker image bernama **sanbercode-project-alpine** telah berhasil dibangun

1. **Konfigurasi Projek Laravel**

Pada proses ini, akan diunduh source code dari projek laravel yang berasal dari [repository Gitlab](https://gitlab.com/muhammadyaqin/laravel5_sample) sebagai berikut

|  |
| --- |
| $ git clone https://gitlab.com/muhammadyaqin/laravel5\_sample src |

Kemudian bersihkan dan lakukan re-instalasi Laravel *dependencies* dengan bantuan **composer**. Sebelumnya, pastikan terlebih dahulu bahwa package PHP telah terinstal di dalam sistem operasi

|  |
| --- |
| $ cd src  $ curl -sS https://getcomposer.org/installer | sudo php -- --install-dir=/usr/bin --filename=composer  $ sudo yum install php72 php72-php-mbstring php72-php-zip php72-php-xml -y  $ rm -rf vendor/ && composer install |

Saat semua *dependencies* telah berhasil ditambahkan, generate sebuah *applicatio*n *key* dan lakukan pengaturan pada **.env** file

|  |
| --- |
| $ mv .env.example .env  $ php artisan key:generate  $ vim .env |
| APP\_NAME=Laravel  APP\_ENV=local  APP\_KEY=base64:KLjVpji+XbQbiebziSXldjS4/RUIvBoZx6zuiCiQxgE=  APP\_DEBUG=true  APP\_URL=http://localhost  LOG\_CHANNEL=stack  DB\_CONNECTION=mysql  DB\_HOST=127.0.0.1  DB\_PORT=3306  DB\_DATABASE=**laravel**  DB\_USERNAME=**devops**  DB\_PASSWORD=**12345** |

**Konfigurasi Virtual Host (Nginx)**

Pada tahap ini, akan ditambahkan sebuah *virtual host*, sebut saja **default.conf**, yang akan digunakan untuk mengatur web server untuk mengeksekusi file **PHP** dan **HTML** dengan **root directory** pada **/var/www/html/app/public**

|  |
| --- |
| $ cd ..  $ vim default.conf |
| server {  listen 80 default\_server;  index index.php index.html;  root /var/www/html/app/public;  server\_name \_;  location / {  try\_files $uri /index.php?$args;  }  location ~ \.php$ {  fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;  fastcgi\_pass 127.0.0.1:9000;  fastcgi\_index index.php;  include fastcgi\_params;  fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;  fastcgi\_param PATH\_INFO $fastcgi\_path\_info;  }  } |

**Konfigurasi PHP-FPM**

Secara default, PHP-FPM:*alpine* tidak akan membuat *sock file* yang melakukan *listening* pada port 9000. Untuk itu, buat konfigurasi baru untuk menggantikan isi dari konfigurasi **www.conf**

|  |
| --- |
| $ vim www.conf |
| [www]  user = www-data  group = www-data  listen = 127.0.0.1:9000  listen.mode = 0660  pm = dynamic  pm.max\_children = 5  pm.start\_servers = 2  pm.min\_spare\_servers = 1  pm.max\_spare\_servers = 3 |

**Konfigurasi Mysql**

Sama halnya dengan PHP-FPM, MySQL pada *alpine* tidak akan membuat *sock file* yang melakukan *listening* pada port 3306. Untuk itu kita buat konfigurasi baru untuk menggantikan isi dari file **my.cnf**

|  |
| --- |
| $ vim my.cnf |
| [mysqld]  user = root  port = 3306  datadir = /var/lib/mysql |

**Konfigurasi Entry Point**

Pada tahap ini, akan dibuat sebuah *script*, sebut saja **startup.sh**, yang berfungsi sebagai **task** **scheduler** yang akan menjalankan layanan **Nginx, php-fpm,** dan **MySQL** serta melakukan inisialisasi database saat **container** hendak dijalankan oleh *container manager*.

|  |
| --- |
| $ vim startup.sh |
| # Initialize Service & Database  mysql\_install\_db  mysqld\_safe &  sleep 5  # Create Database  mysql -e "CREATE DATABASE ${MYSQL\_DATABASE} /\*\!40100 DEFAULT CHARACTER SET utf8 \*/;"  # Create additional mysql user with privileges  mysql -e "CREATE USER ${MYSQL\_USER}@localhost IDENTIFIED BY '${MYSQL\_PASSWORD}';"  mysql -e "GRANT ALL PRIVILEGES ON ${MYSQL\_DATABASE}.\* TO '${MYSQL\_USER}'@'localhost';"  mysql -e "FLUSH PRIVILEGES;"  # Replace default .env file with new db-name, username & password  sed -i "s/DB\_DATABASE=homestead/DB\_DATABASE=$MYSQL\_DATABASE/g" .env  sed -i "s/DB\_USERNAME=homestead/DB\_USERNAME=$MYSQL\_USER/g" .env  sed -i "s/DB\_PASSWORD=secret/DB\_PASSWORD=$MYSQL\_PASSWORD/g" .env  php artisan migrate  php-fpm -D; nginx -g "pid /tmp/nginx.pid; daemon off; |

**Re-build Dockerfile**

Setelah itu, kita perbaharui **Dockerfile** untuk menambahkan **projek laravel**, **config file** serta **entry point**

|  |
| --- |
| $ vim Dockerfile |
| FROM php:7.2-fpm-alpine  RUN apk add --update mysql mysql-client nginx  RUN docker-php-ext-install mysqli  RUN docker-php-ext-install -j$(nproc) pdo\_mysql  **RUN mkdir /run/nginx**  **RUN mkdir /run/mysqld**  **RUN rm -f /var/cache/apk/\***  **COPY --chown=82:www-data src /var/www/html/app**  **COPY default.conf /etc/nginx/conf.d/default.conf**  **COPY www.conf /usr/local/etc/php-fpm.d/www.conf**  **COPY my.cnf /etc/my.cnf**  **COPY startup.sh /startup.sh**  **ENV MYSQL\_DATABASE=laravel**  **ENV MYSQL\_USER=devops**  **ENV MYSQL\_PASSWORD=12345**  **WORKDIR /var/www/html/app**  **ENTRYPOINT ["sh", "/startup.sh"]**  **EXPOSE 80** |

**Konfigurasi Docker Registry**

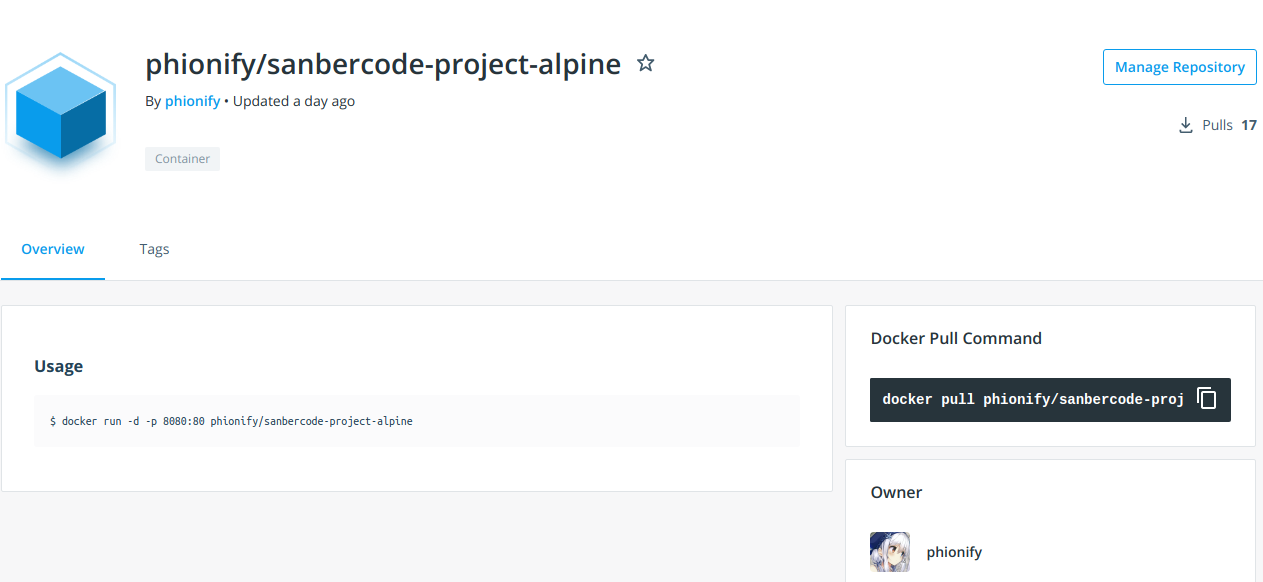
Terlebih dahulu lakukan registrasi pada [Docker Hub](http://hub.docker.com) untuk memperoleh akses ke *docker registry*. Kemudian melalui *command line* kita lakukan otentikasi menggunakan perintah **docker login** dengan username yang telah dibuat sebelumnya

|  |
| --- |
| $ docker login -u **phionify** |
| Password:  Login Succeeded |

Selanjutnya rebuild kembali **Dockerfile** dengan struktur nama **username**/**nama\_image**. Kemudian push Docker Image menuju ke Docker hub.

|  |
| --- |
| $ docker build -t **phionify/sanbercode-project-alpine .**  $ docker push **phionify/sanbercode-project-alpine** |

Hasilnya, image siap diakses di <https://hub.docker.com/r/phionify/sanbercode-project-alpine>



1. **Kubernetes Environment Setup**

Pada tahap ini, terlebih dahulu pastikan bahwa layanan kubernetes serta kubernetes cluster telah diatur sebelumnya. Jika belum, lakukan instalasi dan konfigurasi sebagaimana yang tertera pada petunjuk [berikut ini](https://docs.google.com/document/d/1n-XVKXprAMbsX1Yrc3qD-DRI25zVE-1L9xDqGmZChfo/edit?usp=sharing)

**Kubernetes Application Deployment**

Untuk dapat membuat deployment object, kita dapat membaginya ke dalam dua cara, yakni menggunakan perintah *kubectl* atau mendefinisikan sebuah YAML file

1. Kubectl Deployment

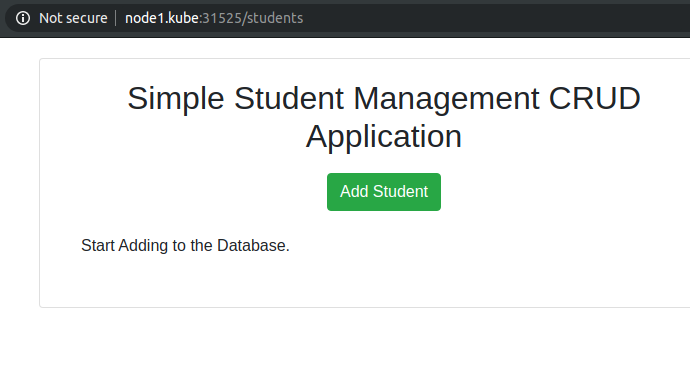
Pada tahap ini, pembuatan deployment dapat dilakukan dengan mudah dengan mengambil **Docker Image** dari docker registry. Akan tetapi, *kubectl* memiliki beberapa kelemahan, yakni kurangnya fitur untuk mengatur konfigurasi lebih lanjut, seperti halnya *persistent volume*, dll. Berikut implementasinya

|  |
| --- |
| $ kubectl create deployment app --image=phionify/sanbercode-project-alpine  $ kubectl create service nodeport app --tcp=80:80 |
| deployment.apps/app created service/app created |

Kemudian, cek apakah pods dan service **app** telah berjalan

|  |
| --- |
| $ kubectl get pod,svc -o wide |
|  |

Hasilnya diketahui bahwa service berjalan pada <http://node1.kube:31525>



1. YAML File Deployment

Pada tahap ini, pembuatan deployment dilakukan dengan mendefinisikan struktur file YAML. Dari sini, dapat dijabarkan berbagai argumen dan komponen secara lengkap. Oleh karena itu, di sini kita akan menjalankan skenario dimana **app** yang dijalankan memiliki sebuah **persistent volume** yang berfungsi sebagai storage system yang menyimpan database dari waktu ke waktu, sekalipun pod telah dihapus.

Pertama-tama, dilakukan pembuatan persistent volume dengan deklarasi sebagai berikut

|  |
| --- |
| $ vim pv.yml |
| ---  apiVersion: v1  kind: PersistentVolume  metadata:  name: pv-volume  labels:  type: local  spec:  storageClassName: manual  capacity:  storage: 2Gi  accessModes:  - ReadWriteMany  persistentVolumeReclaimPolicy: Retain  hostPath:  path: "/mnt/data/mysql" |

Di sini, dapat terlihat bahwa dialokasikan sebuah storage sebesar **2Gb** dengan *local hostPath* pada **/mnt/data/mysql**. Selanjutnya apply konfigurasi tersebut dengan perintah

|  |
| --- |
| $ sudo mkdir -p /mnt/data/mysql  $ sudo chown -R nobody:nogroup /mnt/data/mysql  $ kubectl apply -f pv.yml  $ kubectl get pv |
| persistentvolume/pv-volume created    NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS  pv-volume 2Gi RWX Retain Available |

Kemudian, tambahkan persistent volume claim sebesar 1Gb sebagai partisi data untuk database MySQL

|  |
| --- |
| $ vim pvc.yml |
| ---  apiVersion: v1  kind: PersistentVolumeClaim  metadata:  name: mysql-claim  spec:  storageClassName: manual  accessModes:  - ReadWriteMany  resources:  requests:  storage: 1Gi |

Kemudian apply konfigurasi tersebut dengan kubectl

|  |
| --- |
| $ kubectl apply -f pvc.yml |
| persistentvolumeclaim/mysql-claim created  NAME STATUS VOLUME CAPACITY ACCESS MODES  mysql-claim Bound pv-volume 2Gi RWX |

Terakhir, tambahkan object *deployment* dan *service* sebagaimana berikut ini

|  |
| --- |
| $ vim app.yml |
| ---  apiVersion: apps/v1  kind: Deployment  metadata:  name: app-deployment  labels:  app: app-server  spec:  replicas: 1  selector:  matchLabels:  app: app-server  template:  metadata:  labels:  app: app-server  spec:  containers:  - name: app  image: phionify/sanbercode-project-alpine  ports:  - containerPort: 80  volumeMounts:  - name: mysql-persistent-storage  mountPath: /var/lib/mysql  volumes:  - name: mysql-persistent-storage  persistentVolumeClaim:  claimName: mysql-claim  ---  apiVersion: v1  kind: Service  metadata:  name: app-service  labels:  app: app-server  spec:  ports:  - port: 80  targetPort: 80  nodePort: 32000  type: NodePort  selector:  app: app-server |

Kemudian apply konfigurasi tersebut dan tunggu beberapa saat hingga pod dan service berjalan

|  |
| --- |
| $ kubectl apply -f app.yml |
| deployment.apps/app-deployment created  service/app-service created |

Selanjutnya, cek apakah service dan pod telah berjalan dengan perintah sebagai berikut

|  |
| --- |
| $ kubectl get svc,pod -o wide |
|  |

Hasilnya service dan pod telah berjalan dan siap diakses pada <http://node1.kube:32000>

