Docker

# Modern Application Concept

Pembuatan aplikasi modern dituntut untuk dapat berjalan di multi platform dan dapat di-test secara individual dan tidak tergantung lagi dengan persyaratan server saat deployment.

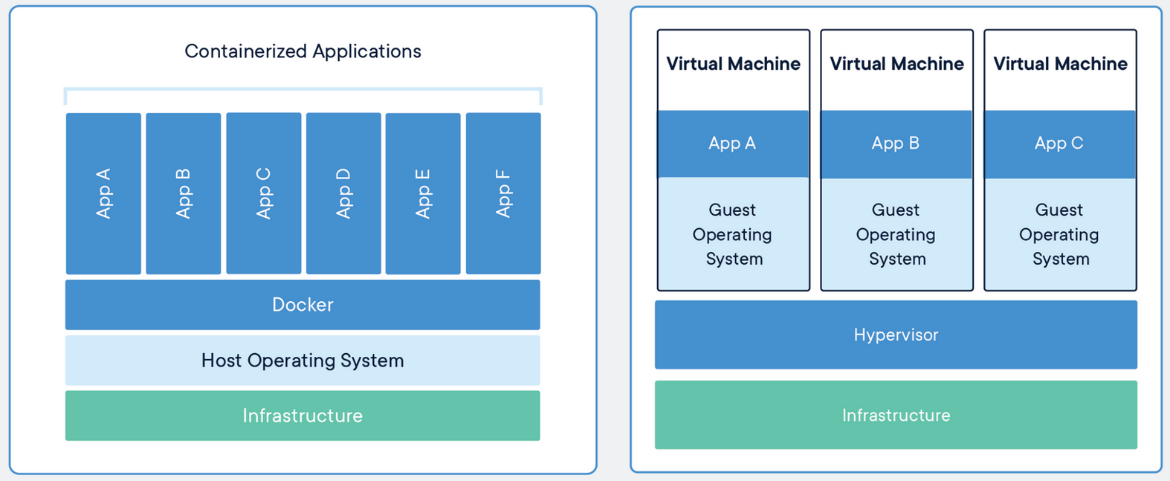
Hal ini dimungkinkan dengan membuat suatu image yg merupakan sekumpulan dll assembly beserta dengan dll pendukung nya yang nantinya akan berjalan di sebuah container.

Docker merupakan sebuah container yg dapat menggunakan image dan menjalankan image tsb dalam container.

## Perbedaan VM dan Container

Kalau kita menjalankan aplikasi di VM, maka aplikasi yg berjalan akan sangat tergantung dengan OS dan server nya karena aplikasi akan langsung berjalan diatas OS, sedangkan kalau kita menjalankan aplikasi diatas container, maka aplikasi akan berjalan tidak tergantung OS nya karena aplikasi berjalan diatas container, dengan demikian aplikasi akan dapat berjalan di semua OS asal mempunyai container.

Sebuah container dapat menjalankan lebih dari 1 aplikasi dan juga aplikasi dapat scale out dalam sebuah container kalau memang dibutuhkan.



## Development cycle

Jadi untuk melakukan development sampai tahap testing untuk sebuah modern application

1. Mengubah Code menjadi Image
2. Menyimpan Image di Repository
3. Menjalankan image dalam sebuah container
4. Melakukan testing aplikasi yg berjalan pada container
5. Kalau ada kesalahan, maka akan mengunah lagi code nya.

Dengan cycle development spt ini, kita berharap image yg sudah kita test akan stabil dan nanti nya image yg kita deploy di server production akan berjalan sama seperti apa yg sdh kita testing di container local kita.

Diagram

Description automatically generated

## Containers, Image dan Repository

Hubungan antara containers, image dan Repository yg ada saat ini

Graphical user interface, diagram

Description automatically generated

# Application Development Step

Untuk membuat aplikasi yg dapat berjalan di container maka kita mempunyai beberapa tahapan

## Create Image

Pertama2 kita harus mengubah coding aplikasi menjadi image, kita memerlukan sebuah file yaitu docker file, dimana di dalam file tsb kita akan mengubah code kita menjadi sebuah image.

### Persiapan menggunakan docker file

Untuk path dan penamaan file

1. Semua folder runtime Library harus diletakkan sejajar dengan file solution
2. Docker File yg digunakan sejajar solution
3. Penamaan docker file Dockerfile.[API Project Name]
4. Buat sebuah file .dockerignore yang berisi listing dari directory atau file2 yg nanti nya akan kita abaikan saat melakukan build application
5. Buat sebuah file NuGet.Config yang akan berisi url dari private nuget yg kita pakai

### Penjelasan Docker File

Docker file berisi beberapa perintah2 yg dipakai untuk melakukan copy resource dan melakukan build dan compile app.

**Semua penamaan alias dan path** didalam docker adalah **case sensitive** karena akan berjalan dalam **container open platform**

1. Persiapan Runtime dan Path di docker, Expose untuk PORT di docker nanti, berikan nama alias **base**

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base

WORKDIR /app

EXPOSE 80

1. Persiapkan SDK dan Path di docker untuk melakukan building API project diberi alias **build**

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build

WORKDIR /src

1. Copy **semua yg sejajar dengan file docker** kedalam **path terakhir yaitu src**

COPY . .

1. Build project API dengan compile release dan simpan kedalam <Working Directory>/app/build
   1. Pada Langkah ini semua nuget akan di-restore

RUN dotnet build " API\_TranScope / API\_TranScope.csproj" -c Release -o /app/build

1. Publish dengan compile release dan simpan kedalam <Working Directory>/app/publish dan diberi alias publish

FROM build AS publish

RUN dotnet publish " API\_TranScope / API\_TranScope.csproj" -c Release -o /app/publish

1. Pakai dari alias base (runtime) sebagai alias final, beri path /app, Copy dari alias publish directory <Working Directory>/app/publish

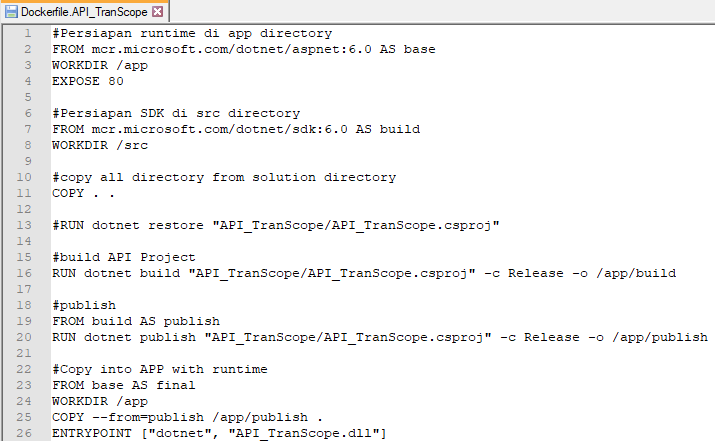
FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

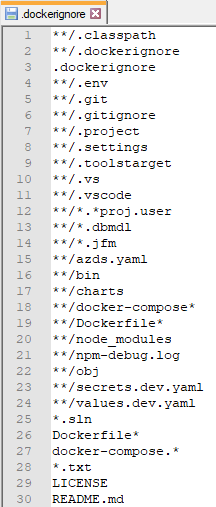
1. Tentukan dll sebagai Entry Point application saat container starting

ENTRYPOINT ["dotnet", "API\_TranScope.dll"]

1. 

### File .dockerignore

Berisi semua listing yang akan diabaikan saat melakukan perintah copy di dalam docker file



### File NuGet.Config

Berisi url private nuget yg kita pakai, juga credential yg dipakai kalau memang nuget tsb membutuhkan credential



Beberapa perintah docker yang dipakai untuk melakukan create image maupun melakukan check image

**Create Image**

1. Persyaratan
   1. Docker File sdh terbentuk

Path

* 1. Nuget config sdh terbentuk

1. Docker command
   1. docker image build -f[docker file name] -t[nama tag dari image] PATH
   2. docker image build -f Dockerfile.API\_TranScope --no-cache=true -t api\_transcope .
   3. Hasil image dengan nama api\_transcope
   4. Sebaik nya penamaan image memakai huruf kecil semua dan pemisah lbh baik menggunakan “-“ bukan “\_”

**Check Image**

Karena image hanya berupa static file, maka kalau kita hendak memeriksanya ada 2 cara

1. Untuk image yg tidak mempunyai runtime command, maka kita dapat melakukan save dalam bentuk tar file
   1. Command
      1. docker image save -o [nama file tar] [id | nama image]
      2. docker image save -o api\_transcope.tar api\_transcope
   2. Nama image dalam case sensitive
2. Image yg mempunyai runtime command dapat kita jalankan dalam sebuah “container sementara” dan bersifat interaction juga entry point nya diarahkan ke bash
   1. Command
      1. Docker run -rm -it -- entrypoint=/bin/bash [nama image]
      2. docker run --rm -it --entrypoint=/bin/bash api\_transcope
   2. -rm untuk melakukan remove container saat keluar dari interarction mode
   3. -it dipakai untuk menjalankan container secara interactive
   4. Setelah command diatas dijalankan, maka
      1. Akan masuk ke dalam docker container di working directory terakhir yaitu app
      2. Untuk melihat file dapat menggunakan perintah ls
      3. Untuk ke directory atas nya memakai cd ..
      4. Untuk melihat directory bawah nya memakai cd <nama directory nya>
      5. Keluar dengan perintah exit

## Run Image di Container

Setelah image terbentuk, maka kita dapat melakukan running container berdasarkan image tsb

1. Command
   1. docker run --name [nama container] port [port OS]:[port container] [nama image]
   2. docker run --name cont\_transcope api\_transcope port 5000:80
   3. port 5000 untuk kita access dari browser

Setelah image berjalan di container, maka kita dapat melakukan checking container

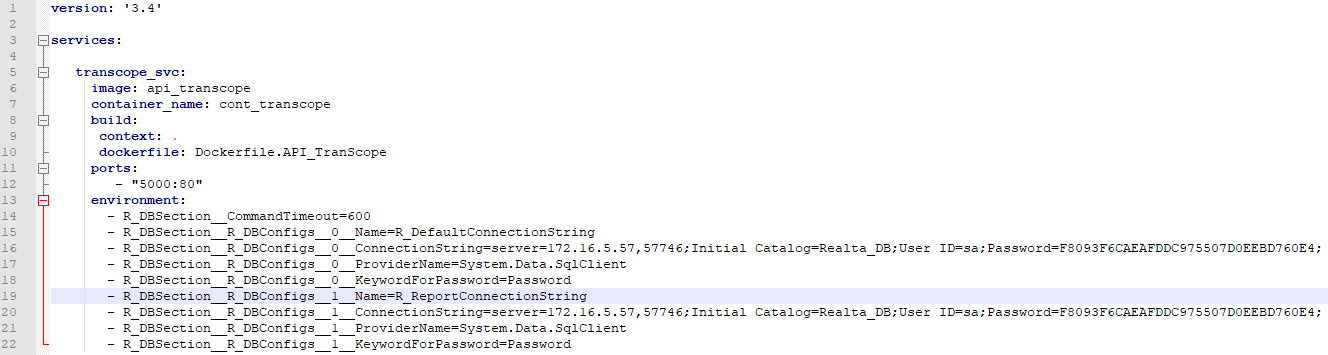
1. Command list container
   1. docker ps -a
   2. melihat semua container dengan status nya
2. Command check port untuk sebuah container
   1. docker port <container id>

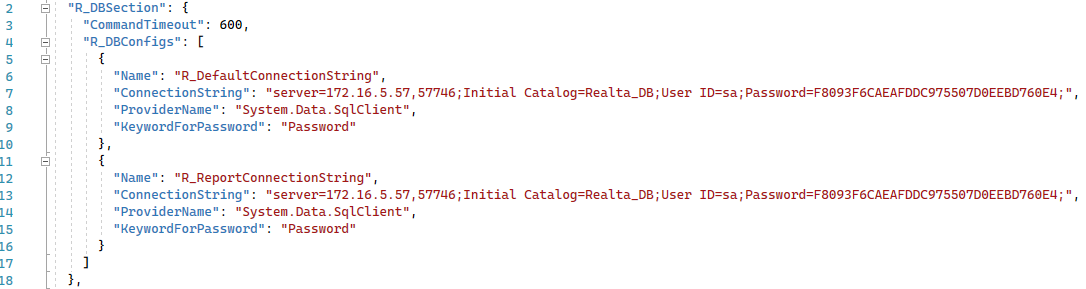
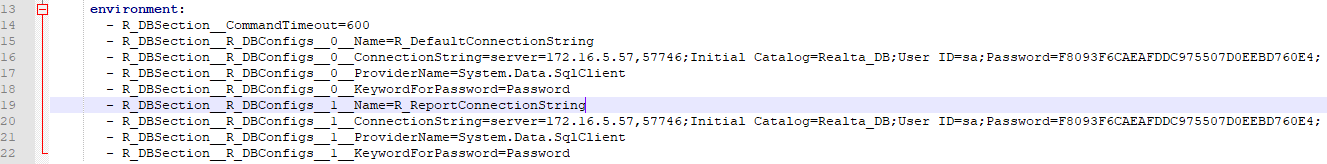
Terkadang saat kita melakukan mapping port, kita tidak mengetahui port mana yg available, jadi sebelum melakukan mapping port, kita dapat melakukan check port available

1. netstat -aon
2. kita akan mendapatkan PID nya yg dapat kita lihat di task manager

## Docker Compose

Docker compose file merupakan sebuah file yg merangkum beberapa service yang akan berjalan dalam container.



1. services
   1. service pertama
      1. nama transcope\_svc
      2. nama image yg dipakai oleh service tsb
         1. api\_transcope
      3. nama container yg dipakai oleh service tsb
         1. cont\_transcope
      4. Section build
         1. context
            1. merupakan path yg menentukan docker file
            2. kalau kosong berarti sejajar dengan file compose ini
         2. dockerfile
            1. nama dari docker file
            2. karena context nya kosong, maka file yg tidak mengandung directory berada sejajar dengan compose file ini
      5. Section ports
         1. Merupakan listing dari mapping ports
            1. 5000:80
            2. Port di container yg di-expose adalah 80 (lihat docker file)
            3. Port dari luar container yg dapat di-akses adalah 5000
      6. Section environment
         1. Section environment ini dipakai untuk meng-override setting yg ada di appsettings.json
         2. Untuk menterjemahkan
            1. setiap pemisahan memakai double dash (\_\_)
            2. Untuk collection harus memakai sequential 0, 1,…..
         3. 
         4. Diterjemahkan
         5. 

Kubernetes

# Kubernetes as containers platform

Kubernetes adalah sebuah platform untuk me-manage container sehingga semua container resources dapat digunakan secara efektif oleh container2 yg sedang berjalan.

Docker akan melihat apakah ada container2 yg sdh tidak terpakai dan apakah ada container2 yg membutuhkan resources, Kubernetes sebagai platform yg akan mengatur semua itu.

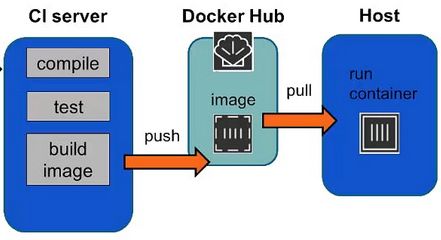
Kubernetes for multi containers orchestration

Graphical user interface

Description automatically generated

## Push Image into Repository

Untuk melakukan deploy ke Kubernetes, maka diperlukan repository image dimana Kubernetes harus menarik image dari repository.



Kita dapat melakukan pemindahan image dari docker ke suatu repository, ada beberapa Langkah

1. Login ke account docker hub
   1. docker login
2. Re-Tag image dengan nama [account name]\[image name]
   1. command
      1. docker tag [nama image] [account]/[nama image]
      2. docker tag api\_transcope dockerguntech/api\_transcope
3. Push image yg sudah di retag ke docker hub
   1. Command
      1. docker push [account]/[nama image]
      2. docker push dockerguntech/api\_transcope

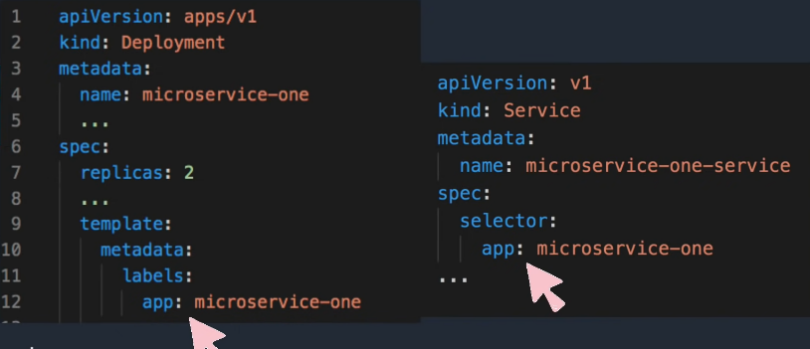
## Container dan Image dalam platform

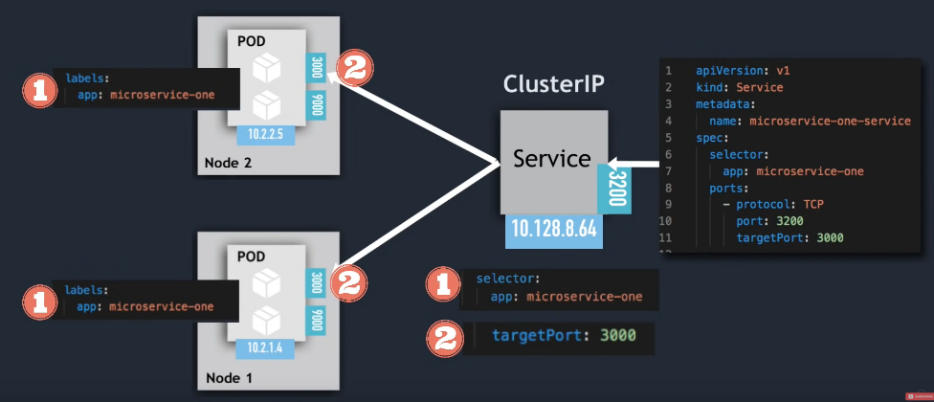
Untuk menggunakan image yg sudah ada di Repository, maka kita memerlukan Kubernetes file.

Dalam Kubernetes file kita dapat membagi menjadi 2 bagian

1. Deployment
   1. Mengatur spec dari deployment
   2. Image akan berjalan dalam container
   3. container akan ada didalam pod
2. Services
   1. Mengatur access dari luar supaya dapat menggunakan container yg ada pada pod

Pada file Kubernetes, hubungan antara bagian deployment dan bagian services





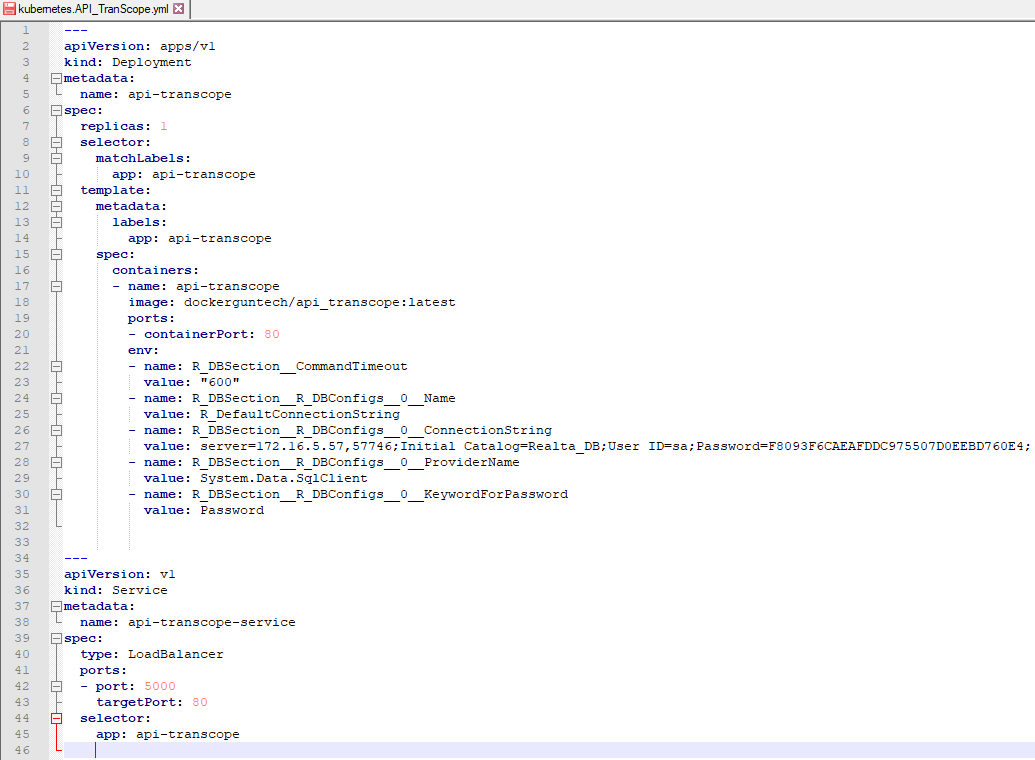
## Kubernetes File structure

### Deployment

1. Penamaan deployment maupun service tidak boleh menggunakan “\_”, maka diganti “-“
   1. api-transcope
2. replica untuk menentukan jumlah instance yg harus available
3. label dan selector menentukan pod2 mana yg termasuk dalam deployment (dalam hal ini pod ada di template bukan di file pod yg terpisah)
4. template merupakan penjabaran dari pod
5. template.labels.app merupakan label app dari pod (label adalah category yg kita buat bisa version dsb-nya)
6. Template.Containers merupakan penjabaran dari container apa saja yg akan masuk dalam 1 pod (1 pod bisa lbh dari 1 container)
7. Didalam containers ada list of container yg akan menandakan
   1. nama dari container
   2. image yg dgunakan dalam container
   3. port untuk container (nantinya akan dipakai oleh service untuk berhubungan dengan container)
   4. env merupakan environment variable yg diperlukan oleh image tsb

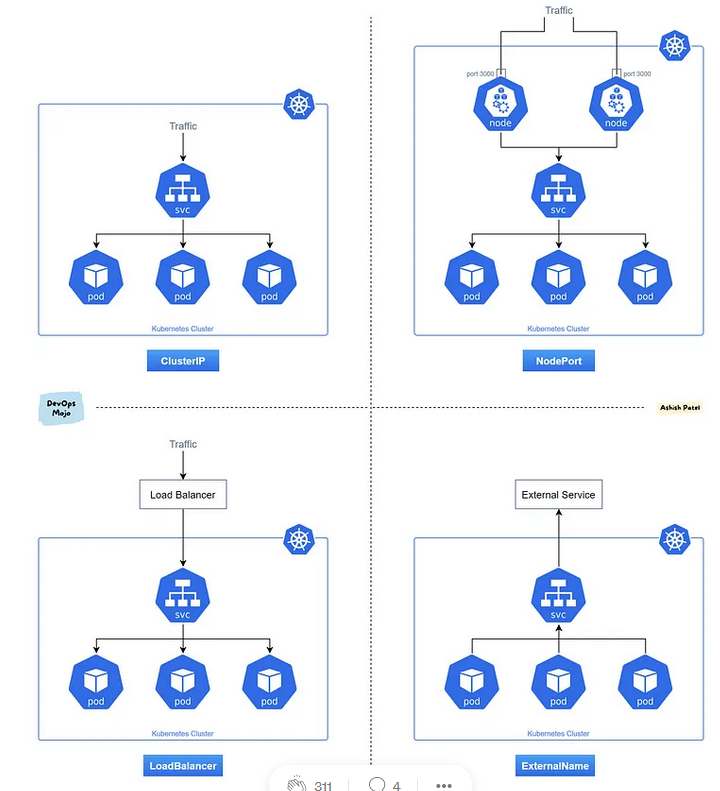
### Service

1. name merupakan nama dari service
2. type – lihat beberapa type service
3. selector dan target Port menentukan service tsb akan menggunakan container dg label tertentu dengan port tertentu
4. Port dipakai untuk meng-ekspose Port yg dapat di-akses dari luar



#### Beberapa konfigurasi Type Service

Type service menentukan kegunaan dari service tsb



Type Service

1. ClusterIP
   1. Default type dari service, dimana service tsb hanya dapat diakses dalam 1 cluster oleh service yg lain dalam cluster tsb
2. NodePort
   1. Merupakan service yg dipakai untuk mengekspose service diluar cluster IP dengan memakai static port
   2. Karena menggunakan static IP, maka harus dipastikan port yg digunakan tidak digunakan oleh port yg lain
   3. Port yg dapat digunakan range antara 30000–32767
   4. Harus diset secara manual
3. LoadBalancer
   1. Merupakan service Load Balancer yg kita expose keluar cluster, sehingga dapat diakses oleh load balancer provider
4. ExternalName
   1. Merupakan service yg dapat digunakan oleh service yg lain didalam 1 cluster, dimana service ini menyediakan service external supaya dapat digunakan oleh service lain didalam cluster tsb, misalkan database external atau api 3rd party
   2. External service yg digunakan disini mempunyai level DNS, misalkan database di azure gysql1.database.windows.net