Development Deployment BackEnd Procedure

Procedure ini dipakai untuk melakukan deployment dari development backend phase sehingga dapat dilakukan testing oleh tester terhadap program2 yang sudah di develop oleh developer

# Development Area

Untuk development Area, tujuan nya adalah

1. Developer dapat melakukan self test terlebih dahulu terhadap program yang sudah dibuat.
2. Developer mempersiapkan semua dll back yg diperlukan untuk melukan deployment di server testing.

## Gambaran procedure

### Development

Develop program back

Ex. GLM000100

Self testing API terhadap program yg sudah dibuat

(Reference By Project)

Lakukan compile terhadap program yg dibuat dan kelompokkan dalam directory

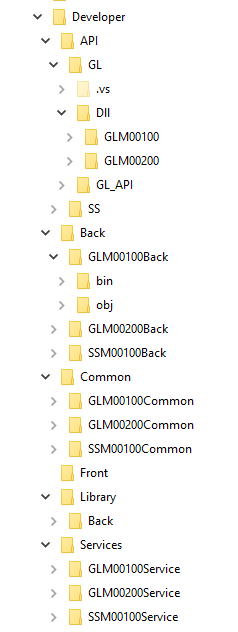
### Persiapan Testing Deployment

Reference assembly dll yg sudah di compile ke dalam API Module

Self testing API terhadap program yg sudah di compile

(Reference By Assembly)

## Directory Structure Developer Area



### Penjelasan directory structure

1. Back
   1. Directory Back dipakai sebagai kumpulan project Back berdasarkan program
2. Common
   1. Directory Common dipakai sebagai kumpulan project Common berdasarkan program
3. Services
   1. Directory Services dipakai sebagai kumpulan project Service berdasarkan program
4. API
   1. Directory API dipakai sebagai kumpulan Module, Misalkan GL, SS
   2. Didalam setiap directory Modul terdiri, misalkan GL, SS
      1. Project API sesuai dengan module
      2. Directory Dll dipakai sebagai kumpulan Dll per program dalam module tsb yg akan di-reference oleh API modul untuk melakukan self testing

### Cara melakukan compile untuk setiap program

1. Procedure ini dilakukan setelah melakukan procedure self testing dengan reference project
2. Buka Command prompt dan masuk kedalam directory Development (Main Back Directory)
3. Lakukan compile terhadap project Service untuk suatu program, misalkan SSM00100Service
   1. dotnet build Services/SSM00100Service/SSM00100Service.csproj -c Release
4. Setelah berhasil maka lakukan publish dll dan hasilnya akan kita tempatkan dalam directory dll yg sudah disediakan di dalam directory API per module
   1. dotnet publish Services/SSM00100Service/SSM00100Service.csproj -c release -o .\API\SS\Dll\SSM00100

( Sesuaikan pada folder tempat akan program disimpan).

* 1. Lakukan pembuangan pada file dengan type .pdb, karena file tersebut merupakan file debug dari filenya.

1. Dalam project API per module, lakukan reference assembly untuk program service tsb
2. Lakukan self testing untuk API program tsb.

# Server Area

Untuk server area tujuan nya adalah

1. Developer melakukan deployment ke server
2. Tester dapat melakukan testing untuk program-program yang sudah selesai di develop oleh developer.

Di bagian server are disediakan structure directory hanya untuk API saja.

## Gambaran Procedure

### Persiapan pemindahan Dll dari Developer Area ke Server Area

Memilih directory API version sesuai dengan versi real-code yang di develop dan di test

Copy-kan Dll per directory program dari Developer area ke directory Dll yg ada di dalam directory per Modul di API version

Lakukan reference assembly di API project sesuai dengan modul nya

### Persiapan Testing API dengan menggunakan Visual Studio

Compile API Modul Project sesuai dengan modul dari program yg di deploy

Lakukan self testing terhadap API program tsb

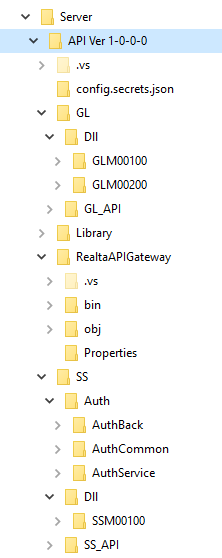
### Persiapkan testing dengan menggunakan Docker dan kubernetes

Jalankan Deployment Docker lalu lakukan test lewat postman

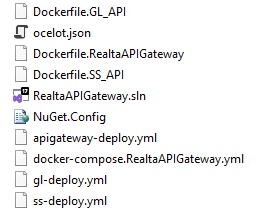
Jalankan Deployment Kubernetes dan lakukan test lewat postman

## Directory Structure dan File Template di API Server Area

### Directory Structure



### File Template di API Server Area



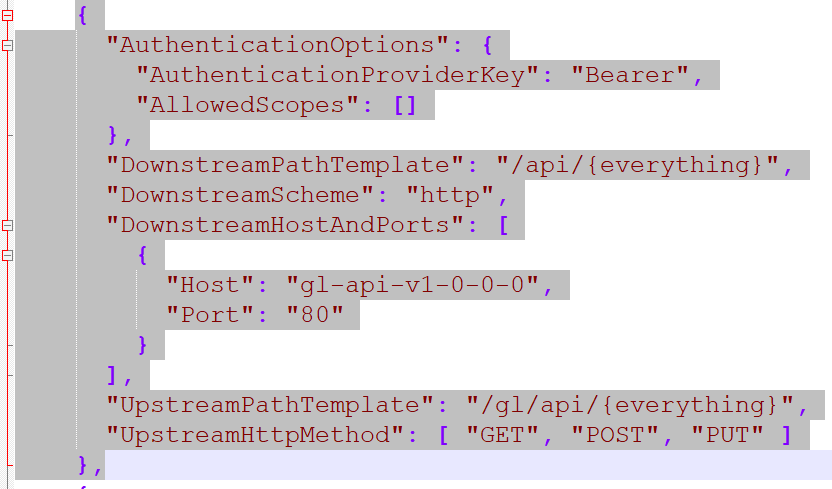
### Penjelasan Directory Structure

1. Directory API dipisahkan berdasarkan version dari API yang akan di deploy
   1. Didalam setiap directory API akan akan mempunyai directory berdasarkan modul nya
   2. Modul dalam directory API version akan terdiri dari
      1. Project API sesuai dengan Module
      2. Directory Dll yg merupakan kumpulan dll per program
         1. Directory per program sesuai dengan Mapping program tsb masuk kedalam module apa.
   3. Project API Gateway
      1. Merupakan project yg dipakai untuk melakukan
         1. Routing
         2. HTTPS terminating
         3. Central Authentication
   4. Directory Library
      1. Merupakan runtime library yg diperlukan untuk semua project API per module

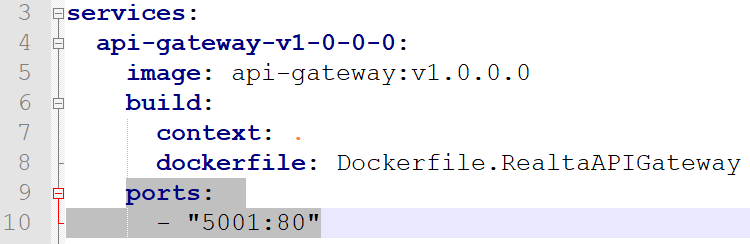
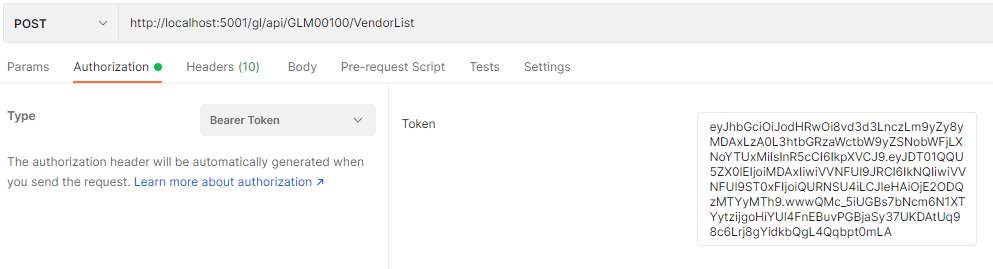
Penjelasan File Template

1. File template ada di dalam masing2 directory API version
2. Docker File
   1. Semua file yg dimulai dengan Dockerfile.XXX
   2. Merupakan file yg dipakai untuk melakukan build image untuk setiap API Module dan Gateway
3. Docker Compose File
   1. Hanya 1 file yg diawali dengan docker-compose.XXXXX.yml
   2. Merupakan File YAML yg dipakai untuk melakukan
      1. Build dan Rebuild Image untuk semua API yg terkait
      2. Run docker container spy kita dapat melakukan testing lewat postman
4. File Ocelot.json
   1. Merupakan file yg dipakai oleh Gateway untuk melakukan beberapa hal
      1. Routing
      2. Authentication
5. Nuget.Config
   1. Merupakan file yg dipakai untuk melakukan Image Build dari API project yg menggunakan private Nuget package
6. Kubernetes Deployment File
   1. Semua file dengan pattern XXX-deploy.yml
   2. Merupakan file YAML yg dipakai untuk
      1. Melakukan deployment untuk setiap API Module dan Gateway ke Kubernetes

### Persiapan Docker Deployment

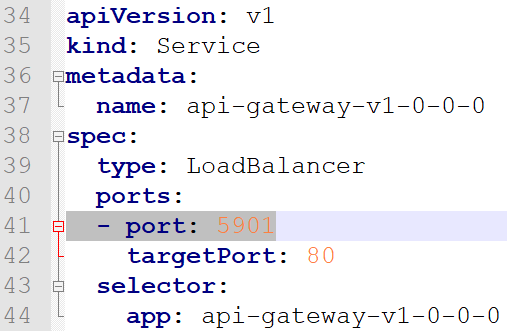
1. Persiapan ini hanya dilakukan sekali saja untuk setiap pergantian API Version
2. Buka File docker-compose.RealtaAPIGateway.yml
   1. Ubah semua pattern version menjadi version yg dikehendaki
      1. Pattern V9-9-9-9
      2. Pattern V9.9.9.9
      3. Dari version asalnya menjadi version yg kita kehendaki
         1. V1-0-0-0 menjadi V1-0-0-1
         2. V1.0.0.0 menjadi V1.0.0.1
   2. Kalau ada penambahan module, maka tambahkan bagian
      1. Panambahan Service
      2. Penambahan depens\_on
      3. 
      4. Tentukan port yg masih tersedia untuk melakukan testing sementara di docker
3. Kalau ada penambahan Module, maka tambahkan docker file untuk module tsb
4. Buka file ocelot.json
   1. Ubah semua pattern version menjadi version yg dikehendaki
      1. Pattern V9-9-9-9
      2. Dari version asalnya menjadi version yg kita kehendaki
         1. V1-0-0-0 menjadi V1-0-0-1
   2. Kalau ada penambahan Module, maka tambahkan bagian Routes
      1. Upstream
      2. Downstream
      3. AuthenticationOptions
   3. 

### Docker Deployment and Test

1. Procedure ini dilakukan setelah melakukan compile dan testing API Module di Visual Studio yg ada di server
2. Buka Command prompt dan masuk ke directory API version yg ada di server
3. Untuk melakukan build hanya sebuah Image API Module, misalkan module GL
   1. Hal ini dilakukan kalau misalkan kita tau perubahan hanya terjadi di module tsb
   2. docker image build -f Dockerfile.GL\_API --no-cache=true -t gl-api:v1.0.0.0 .
4. Untuk Build Image semua module
   1. docker-compose -f docker-compose.RealtaAPIGateway.yml build
5. Untuk menjalankan docker container
   1. docker-compose -f docker-compose.RealtaAPIGateway.yml up
6. Lakukan testing dengan postman sesuai dengan port yg ada di api-gateway service
   1. Dalam contoh ini 5001
   2. 
   3. Contoh postman
      1. <http://localhost:5001/api/Auth/Login>
         1. Untuk mendapatkan Token
      2. Selanjutnya token harus diisikan di bagian Authentication sebelum melakukan testing untuk program2 di API Module
         1. 
      3. Contoh API program
         1. <http://localhost:5001/gl/api/GLM00100/VendorList>
         2. <http://localhost:5001/gl/api/GLM00200/EmployeeList>
         3. <http://localhost:5001/ss/api/SSM00100/CustomerList>
7. Untuk men-delete docker container deployment
   1. docker-compose -f docker-compose.RealtaAPIGateway.yml down

### Persiapan Kubernetes Deployment

1. Persiapan ini hanya dilakukan sekali saja untuk setiap pergantian API Version
2. Create ConfigMap
   1. Untuk melakukan load file configuration yg dibutuhkan
   2. Untuk memberikan nama pada configMap sesuai version dengan pattern
      1. gateway-configs-v9-9-9-9
   3. Check terlebih dahulu ConfigMap yg ada
      1. kubectl get configmap
   4. Kalau sudah ada, maka delete dahulu configmap nya
      1. kubectl delete configmap gateway-configs-v1-0-0-0
   5. Create ConfigMap
      1. kubectl create configmap gateway-configs-v1-0-0-0 --from-file=./ocelot.json
   6. Untuk melihat isi dari ConfigMap yg kita create
      1. kubectl describe cm gateway-configs-v1-0-0-0
3. Buka semua file deployment XXX-deploy.yml
   1. Ubah semua pattern version menjadi version yg dikehendaki
      1. Pattern V9-9-9-9
      2. Pattern V9.9.9.9
      3. Dari version asalnya menjadi version yg kita kehendaki
         1. V1-0-0-0 menjadi V1-0-0-1
         2. V1.0.0.0 menjadi V1.0.0.1
4. Kalau ada penambahan modul, tambahkan file deployment untuk module tsb
5. Ocelot.json sudah diubah pada topik [Persiapan Docker Deployment](#_Persiapan_Docker_Deployment)
6. Sesuaikan port yg ditentukan untuk versi tsb di gateway deployment file yaitu apigateway-deploy.yml bagian service Load Balancer



### Kubernetes Deployment and Test

1. Buka Command prompt dan masuk ke directory API version yg ada di server

Kita bias liat di KUBERNETESNOTE.txt,

Letaknya pada computer server

1. Sign In Docker Hub
   1. docker login -u realtadev -p P@ssw0rd25A

//hal ini digunakan pada computer server sedangkan saat mau mencoba pada pc sendiri, gunakan aku login HUB sendiri

1. Re-Tag semua docker Image module dan gateway sesuai dengan versinya dengan prefix sesuai dengan account hub realta/xxxxx, misalkan
   1. docker tag ss-api:v1.0.0.0 realtadev/ss-api:v1.0.0.0
2. Push semua Image per module dan gateway sesuai dengan versinya ke Docker Hub
   1. docker push realtadev/ss-api:v1.0.0.0
3. Deploy Kubernetes untuk semua Deployment file
   1. kubectl apply -f ss-deploy.yml
4. Lakukan testing API sesuai dengan port deployment di file apigateway-deploy.yml
   1. <http://localhost:5901/api/Auth/Login>
5. Untuk menghapus deploymeny yg sudah tidak terpakai
   1. kubectl delete -f ss-deploy.yml