Nama : Dhea Safira

No BP : 2201081005

Prodi : D3 Teknik Komputer

Jurusan Teknologi Informasi

Mata Kuliah : Pemrograman Microservice

- **Belajar Microservices -7 Remote Procedure Invocation**

  Salah satu cara berkomunikasi di dalam microservices. Yang pertama namanya Remote prosedur in Vocation atau ada juga yang sering bilangnya
  remote prosedur call. Tapi intinya sama aja sih ya kita bahas. Misal aja kita ada kasus seperti ini ya contohnya di sini. Misal kita di sini punya order servis, di sini juga ada product servis dimana
  order servis ini adalah servis yang menyimpan data penjualan atau pemesanan. Ya anggap aja kayak di toko on line itu ya data penjualannya atau pesanannya ada di dalam order servis. Nah, selanjutnya di sini ada product servis. Produk servis itu adalah data semua produk atau katalog yang kita jual.
  Biasanya saat kita ingin melakukan order atau bikin order atau penjualan atau pemesanan, udah pasti kita akan butuh data produk yang akan di pesan. Ya, di sini adalah butuh data product. Pertanyaannya, bagaimana cara
  makan servis ini oder servis untuk mendapatkan data si prodaknya? Nah sebelumnya kan kita udah bahas ya tentang sharing database. Nah tapi anggap aja sekarang kita udah bikin MySQL servisnya secara ideal ya enggak lagi dalam masa transisi dari monolitik ke masa servis. Tapi anggap aja ini bikinnya udah ideal.
  Nah, apa rekomendasinya? Kalau kita memang butuh data dari tempat lain. Nah, kalau misalnya kita emang butuh data dari servis lain, maka yang harus kita lakukan adalah. Sebenarnya kita bisa menggunakan salah satunya adalah remote prosedur infection. Jadi komunikasikan
  ter servis. Idealnya komunikasi dilakukan melalui r, p, e ya, atau remote. Prosedur infection atau sekak sebenarnya yang paling banyak orang tahunya atau familiar ya itu namanya revisi remote prosedur call di. Tapi semuanya intinya sama aja ya. Jadi kalau ada yang bilang remote prosedur in vocation
  atau juga remote prosedur call itu intinya sama aja. Nah, jadi seperti yang tadi sudah dibahas, jadi tidak direkomendasikan komunikasi dilakukan via database. Jadi nggak ada lagi tuh istilahnya sering database. Apalagi kalau tiba tiba kasusnya seperti tadi in di sini, post Gray ini di sini MongoDB. Nggak lucu kan? Di sini
  order servisnya Karena dia bukan MongoDB, akhirnya dia harus konfigurasi di sini database MongoDB karena dia butuh nembak ke database product servis ini. Jadi itu nggak direkomendasikan. Yang direkomendasikan adalah menggunakan remote prosedur infection atau remote prosedur call
  Nah, ini apaan sih remote Prosedur infection? Nah, sederhananya dia itu. Kalau sekarang itu mungkin teman teman tahunya bikin api. Iya. Nah jadi ini juga sama jadi bikin api air yang

nanti di pake oleh servis lain. Nah pertanyaannya bikin api hanya gayam? Gimana caranya? Nah, sebenarnya ada banyak
sekali saat ini cara membuat remote prosedur infection atau remote prosedur call atau yang kekiniannya adalah api air itu banyak banget sekarang. Contoh contoh yang saat ini ini ada beberapa daftar yang saat ini populer. Yang pertama yang paling popular adalah res poll API. Iya, jadi pake http
Nah ini saat ini paling populer banget. Kenapa? Karena paling gampang dibikinnya. Hampir semua bahasa pemograman sekarang udah support yang namanya HTTP, jadi harusnya ini paling gampang dan paling mudah. Nah selanjutnya yang sekarang lagi naik daun contohnya adalah g RPC. Ya, ini
saya kurang tau ya dibuatnya oleh google atau bukan. Tapi yang pasti ini lagi popular banget di bahasa pemograman Golang dan sekarang udah mulai merembet sih ke bahasa pemograman yang lain. Udah banyak yang pake ge er p si cuma paling mudah yang mana paling mudah tetap ada. Tepai sebenarnya karena ge er p si teman teman harus
bikin speknya abis itu di kopel jadi bahasa kelayan yang teman teman pakai. Nah, atau ada juga pendahulunya sebelum ge er visi populer itu ada juga. Apa aja? Trip ini kurang lebih hampir mirip dengan ge er PC ya. Jadi intinya nanti teman teman bikin spek HP Aie abis itu di jendela chat bisa jadi java
Kelayan atau Java Server atau bahasa pemograman yang lainnya. Ini sama nih, kedua ini. Nah selanjutnya yang ketiga ada SOAP. Nah ini kalau misalnya temen temen istilahnya programmer jadul ya programmer lama itu pasti udah kenal dengan yang namanya SOAP ini. Simpel Object Akses Protocol
ini kalau jaman dulu istilahnya adalah webservice. Jadi kalau webservice itu dulu itu selalu pakai SOAP. Kalau sekarang kan kalau bikin web servis biasanya kita pakenya rest poll ya. Tapi kalau dulu itu pakenya soap dan ini dulu popular banget dan harusnya di korporat korporat gede ini masih populer terutama
bank ya. Kalau teman teman perhatikan integrasi sama bank itu rata rata mereka nyediain api apinya itu berupa SOAP. Atau kalau sama teman teman di Java itu kenal dengan yang namanya Java RMI Remote Method Infection. Jadi semuanya agak aga mirip sih. Pokoknya remote remote prosedur
atau remote method dan sebagainya. Atau yang paling jadul lagi, ini nih namanya korban Kamen. Objek request broker arsitektur ini dari singkatan sih semuanya keren sih, cuma ini teknologinya jadul banget dan sekarang kayanya udah jarang banget sih yang pake korban ini ya. Nah, kalau temen temen programmer yang lawas
itu yang jadul banget itu harusnya tahu tentang korban dan semuanya masih banyak yang lainnya. Kalau rekomendasi saya gimana? Rekomen saya itu pake yang paling banyak digunakan. Contohnya yang paling banyak digunakan adalah HTTP. Jadi yang pake dulu yang paling baik digunakan. Biar apa? Biar enggak terlalu spesifik ke teknologi. Contohnya kalau HTTP kan hampir semua bahasa pemograman bisa tuh. Kalau saya gak revisi ya mungkin harus dilihat dulu siapa yang support biar revisi. Enggak semua bahasa pemograman support, termasuk Apache trip juga sama. Enggak semua bahasa pemograman support. Kalau SOAP harusnya swap itu semua bahasa pemograman support cuma bahasa pemograman pemograman yang istilahnya enggak
enggak ditargetkan untuk enterprise itu ya itu biasanya agak sulit untuk parsing data suapnya. Nah ini popular banget di bahasa bahasa yang enterprise kayak Java atau si SAP dan teman temannya. Nah, kemarin saya adalah pake yang paling populer dulu. Contohnya saya HTTP atau kalau teman teman emang

Microsoft bisanya pake satu bahasa pemograman ya Teman teman bisa pakai misalnya Giver revisi atau Apache trip. Nah, jadi kesimpulannya, setelah tadi kita tahu nih kalau misalnya kita mau mengekspos sebuah sebuah data atau sebuah istilahnya sebuah API, iya, kita bisa membuat
remod prosedur infection. Nah contohnya misal aja tadi kasusnya adalah order servis butuh data di produk ini. Jadi di produk servis sekarang di sini bikin api misalnya menggunakan respon api air http. Nanti ketika order servis ini butuh data product nya, dia cukup mengengkol
api air ke product servis. Jadi sederhana ini enggak ada dari on the servis itu direct langsung ke mongo alibinya. Nah dengan begitu kalau di sini ternyata dia butuh join beberapa tabel, enggak perlu dilakukan. Cukup kok satu api air dan nanti produk servisnya yang akan nge join beberapa tabel dan mengembalikan resultnya
sesuai dengan yang dibutuhkan. Sama sih ordernya. Keuntungannya apa sih? Menggunakan remote prosedur dan vaksin ini yang pertama adalah sederhana dan mudah. Ya jadi gampang banget itu kaya yang kolom method biasa aja di kode program kita ya jadikan kode kode kode program kita
Kita kan bisa manggil method gitu ya lain juga sama itu mirip mirip kaya manggil method aja, cuma berupa misalnya remote prosedur infection. Nah, selain itu RPA ini biasa digunakan untuk komunikasi request reply. Artinya apa? Ada request yang meminta ada reply
yang menjawabnya. Jadi biasanya dua, bukan dua arah ya, yang biasanya adalah request dan respon. Lah istilahnya kaya gitu ya, jadi selalu ada responnya. Nah selanjutnya biasanya digunakan untuk processing. Processing itu artinya dia butuh nunggu jawabannya. Contohnya kalau
sebelumnya di sini okay, ketika kita create order, dia nembak ke sini terus bilang saya mau nge get order dengan ID 1. Nah sih order servis dia nunggu jawabannya. Jadi nanti setelah ini menjawab baru programnya akan dilanjutkan. Jadi biasanya komunikasi
RP ini digunakan untuk proses yang sing yang menunggu jawaban. Biasanya kolom saya prosesnya asing. Itu biasanya nggak pake HP ya pakai mekanisme cara berkomunikasi lain. Nanti kita akan bahas soal cara yang lainnya. Nggak jadi seperti itu. Jadi sekarang mulai sekarang kalau teman teman saya mau mengekspos sebuah API itu, silakan gunakan remote prosedur infection. Jadi nanti servis yang lain yang membutuhkan datanya nggak langsung direct ke database, tapi dari nembaknya atau aksesnya melalui si LPH ini.

- **Belajar Docker untuk pemula -08 Container(video ke-8)**

Sebelumnya kita udah bahas tentang Remote Prosedur Infection dan sekarang kita akan bahas tentang Messaging. Cara yang kedua yang bisa kita gunakan untuk berkomunikasi antar microservices. Okey, sekarang contoh kasus seperti ini ya. Sebelumnya kan bahas tentang order yang butuh data produk itu bisa menggunakan
remote prosedur infection. Nah sekarang kita akan membuat sempel kasus, misalnya disini ada sempel kasus yang lebih kompleks. Jadi disini ada misalnya order servis bisa dimana? Di sini menggunakan database post grade. Nah misal aja dalam kasus ini ketika kita selesai membuat pesanan atau penjualan itu kita ingin mengirim email ke si customernya dan juga mengirim sms dan juga mengirim data penjualannya ke servis finance dan juga ke service report untuk dibuat nanti reportnya. Nah apa yang terjadi?

Nah ya, jadi bisanya data order itu butuh komunikasi dengan empat servis ini nih. Jadi tadi kan cuma satu, sekarang ada empat sekaligus. Nah, apa yang terjadi kalau kita implementasikan remote prosedur infection? Nah, ini yang kita lakukan kalau kita mengimplementasikan remote prosedur infection ya. Jadi simpel aja sih, semuanya dari order servis. Saat kita pengen ngirim email, tinggal kirim ke email servis, saat butuh sms, kirim ke SMS servis dan selanjutnya tinggal kirim jual ke finance data ordernya dan juga kirim ke report data si ordernya
selesai. Tapi ini memang sangat gampang sekali di implementasikan, karena yang tadi saya bilang ya di RP itu yang sebelumnya saya pernah bahas itu sangat mudah untuk di implementasikan ya. Kalau untuk menggunakan nearby ini, cuma kalau diperhatikan akan ada beberapa problem ketika kita mengimplementasikan RPC
secara masif ya. Kalau misalnya kita seenaknya mengimplementasikan RP, nah ini akan banyak. Ada beberapa hal yang bukan berapa. Jadi ada kasus dimana akan terjadi masalah kalau kita menggunakan remote prosedur infection. Nah, di sini ada masalahnya nih. Nah, akan saya list di slide selanjutnya
Nah, ini masalahnya. Yang pertama adalah pada kasus sebelumnya akan terjadi proses yang sangat lama ketika pembuatan order. Kenapa? Karena akibat dari manggil si email servis dan juga SMS servis. Nah, kalau temen temen udah pernah ngerasain gimana lamanya ngirim email menggunakan, kalau misalnya manual ya itu butuh waktu, misalnya kayak 3 atau 5 detik gitu atau mungkin kurang masa mas juga. Anggap saja lama gitu ya agak lama gitu. Nah, problemnya adalah kalau di sini misalnya nangkapnya di sini 5 detik, di sini 2 detik gitu ya. Nah, sekarang untuk running proses ini butuh 7 detik.
Nah, bayangin dari depan itu ya dari sisi customer ketika nggak bikin pembelian atau penjualan, order baru gitu ya. Nah, dia nunggu 5 datanya 7 detik nih sampai prosesnya selesai. Kenapa? Karena ternyata di belakangnya kita nembak sinyal servis dan juga sama servis. Dan akhirnya respons tim si order servisnya
itu malah makin lemot. Jadi yang tadinya di sini cepet banget karena manggil 2 servis ini akhirnya kena impactnya nih. Karena yang 2 ini agak sedikit lama ini ikutan menjadi lama responnya. Nah selain itu ditambah lagi dengan tambahan respon time finance dan juga report
Anggap aja ini cepet ya 1 detik 1 detik, tapi jadinya akhirnya ini 5, ini 2 in 1 1 jadinya 9 detik. Jadi setiap ada order baru butuh respon itemnya 9 detik dan itu harusnya lumayan lambat. Harusnya itu order. Sorry beken itu kalau mau profit. Respon itu harusnya bisa di bawah 2 sekon. Kalau di atas 2 sekon harusnya itu bisa di bilang lambat. Nah selain itu kalau diperhatikan mengirim datanya bisa berkali kali dengan data yang sama. Contohnya adalah di finance, servis dan report servis. Jadi data order di sini itu dikirim ke finance servis dan juga dikirim ke repo servis, padahal datanya sama. Nah, kejadiannya lagi kalau tiba tiba di sini ada servis baru ini yang butuh data order, maka kita akan mengirim data yang sama ke order yang berbeda. Jadi satu data order itu bisa dikirim ke 3 servis yang berbeda. Padahal datanya sama nih. Nah ini lumayan problem juga. Nah
selain itu, oke deh kalau tadi lambat, misalnya kita ada ide untuk membuat prosesnya menjadi pararel, ya kalau kita buat jadi pararel itu otomatis akan menjadi kompleks dan ribet banget. Bahkan beberapa bahasa pemograman yang nggak support pararel mungkin agak susah
kita yang implementasinya. Jadi implementasi pararel itu kalau bisa sih dihindari. Kenapa? Karena lumayan agak kompleks untuk channelnya. Jadi misalnya ngirim keempat servis ini

dilakukan secara pararel. Nah itu agak sedikit kompleks ya. Contohnya misal aja nih, misal di sini ada payment servis dimana
kalau payment servis ini sukses, baru saya ngirim email dan sms kalau tadi dilakukan pararel. Tiba tiba email dan sms nya ke kirim. Padahal mungkin ketika ngocol si Paiman servis di sini di gagal ya. Jadi ada kasus seperti ini gimana nggak bisa langsung dijadiin pararel sih, prosesnya gitu. Nah, trus solusinya
apa nih? Nah, solusinya adalah ada cara komunikasi lain selain remote Prosedur infection, yaitu adalah messaging. Jadi komunikasi dengan cara messaging sederhananya adalah komunikasi yang dilakukan untuk proses asing. Kronos. Jadi nggak semua
komunikasi itu cocok menggunakan messaging ya. Kalau komunikasinya kaya request, abis itu butuh replay, butuh respon, maka cocoknya menggunakan remote prosedur infection. Tapi kalau diperhatikan di kasus ini, contonya ngirim data ke finance itu ya semua kita nggak peduli gitu ya. Dan respon dari finance yang
penting sih order ngirim data ke finance, termasuk ini. Ini ngirim data ke report, jadi dimana? Enggak peduli juga repotnya kayak gimana, yang penting si order berhasil ngirim datanya ke report, termasuk email email juga. Semuanya gagal atau sudah semuanya. Nggak terlalu peduli sih waktu saya disini, yang penting kirim. Tolong kirimin
email untuk order. Ini juga sama, tolong kirimin sms untuk order misalnya. Jadi semuanya. Semua respon ini nggak terlalu peduli nih datanya, karena yang paling penting adalah ordernya itu dicoba dikirim ke semua servis yang empat ini. Nah, artinya kalau kasusnya seperti ini
harusnya messaging itu lebih cocok di implementasikan di empat integrasi komunikasi servis yang tadi. Jadi dia nggak butuh replay dan sebenarnya bisa dilakukan secara asing semuanya ya jadi asing itu artinya komunikasi dilakukan tanpa harus menunggu proses selesai. Jadi nggak perlu nunggu
si emailnya berhasil terkirim dulu. Nggak perlu nunggu si SMS nya berhasil. Kirim dulu semua. Harusnya itu semuanya nggak perlu ditungguin. Jadi cukup kirim udah selesai, langsung balikin si responnya ke user. Karena yang penting data ordernya udah berhasil disimpan di database postgres nya. Nah, dalam masing kadang
tidak perlu peduli balasan dari servis yang tujuh yang tadi saya bilang ya kalau saya yang pentingkan ordernya udah disimpen di database. Kalau saya kirim email tenyata emailnya gagal. Ya udah la ya nggak perlu di rollback itu ordernya yang penting jadi nggak terlalu pedulilah sama responnya. Termasuk yang tadi yang empat semuanya. Si order itu nggak terlalu perduli dari
saya. Respon dari si 4 servis yang tadi. Nah, biasanya komunikasi messaging membutuhkan message channel. Nah ini yang perlu penting nih. Nanti kita akan lihat detailnya sebagai jembatan untuk mengirim dan menerima data. Jadi biasanya dalam messaging itu data dikirim ke sebuah tempat yang namanya
message channel. Maksudnya itu anggap aja kalau di database itu kaya tabel ya, jadi datanya disimpan di tabel. Nanti kalau mau ngambil datanya itu ngambil dari tabelnya atau ngambil dari message channelnya. Nanti di gambar diagramnya mungkin temen temen akan lebih kebayang. Nah selain itu direkomendasikan
menggunakan aplikasi medsos broker untuk melakukan manajemen message channel. Nah, jadi kalau ini dianggap sebagai tebel ya message channel itu, maka medsos blocker adalah databasenya. Beta base manajemen sistemnya lah ya cocoknya kalau di sini tabel di sini misalnya masalah broker. Contohnya

MySQLnya lah gitu ya kayak gitu. Nah untuk lebih jelasnya kita lihat di gambar. Jadi yang tadi arsitekturnya kita ganti karena kebutuhannya adalah semuanya. Prosesnya dibutuhkan asing dan juga enggak peduli responnya. Nah, jadi nanti kita menginstall aplikasi medsos broker di sini
lalu kita akan buat message channel di dalamnya. Contohnya di sini ada masa channel untuk email, ada masa channel untuk sms, dan ada masa channel untuk order. Nah nanti shoulder to service itu tugasnya nggak perlu langsung nembak ke servis servis yang empat ini. Jadi cukup ngirim data email ke
message channel ini, nanti si email akan menerima secara asing Cronus tanpa harus kita membuat prosesnya secara pararel manual. Jadi ini automatik udah langsung asing kronos. Termasuk saat ngirim sms ke sini ya nanti akan diterima oleh si SMS server secara asing. Kronos
juga. Dan yang terakhir si folder servis cukup ngirim sekali doang ke message channel order. Nanti message channel order ini bisa diterima oleh report servis dan juga finance servis. Dua duanya secara asing kronos dan pararel. Dan nanti kalau tiba tiba oke
ternyata ada servis lagi yang butuh data order tinggal di sini. Saya di sini payment servis misalnya. Ya nanti tinggal konekkan ke masa channel ini dan otomatis ketika ada order masuk ke sini ke masa channel ini, ini akan diterima oleh 3 si servis tersebut. Jadi si OTA servis bahkan sekarang nggak peduli lagi
siapa yang bakal nerima data ini. Jadi kalau misalnya ada siapa pun yang nerima datanya, silakan ambil di message channelnya. Ini formatnya semuanya nggak kayak tabel ya. Ini lebih ke kaya log mungkin. Jadi log itu misalnya first in, first out, dan juga ketika udah di konsumsi
tu datanya ngilang dari masa channelnya. Jadi kalau dari konsumen repot repot nggak kan nerima lagi order yang sama. Termasuk juga kalau di udah di konsum sama finance finest nggak akan nerima data yang sama. Jadi kaya sekali konsumen itu langsung hilang datanya. Nah
ini adalah contoh contoh kasus broker yang saat ini banyak ada di Populer lah istilahnya. Yang pertama yang paling sederhana adalah Redis. Redis itu ada fitur yang namanya Fab, SAP publish, scrap. Nah ini bisa teman teman gunakan sebagai pengganti, bukan sebagai pengganti ya, tapi sebagai
meses brokernya. Untuk manajemen channel masa channelnya atau yang saat ini lagi populer adalah contohnya Apacer Kafka ini lagi popular banget nih, dan saya itu ada juga yang versi lebih ringannya. Untuk meses broker ada namanya Rabit M View atau ada juga yang lagi populer juga nih
namanya NS Kio atau saya. Kalau teman teman pakai pengen pakai solusi cloud computing ya ada juga Google, Pap, SAP dan juga kemajuan webservice SQS. Dan kalau saya teman teman cari lagi soalnya ada banyak banyak banyak banget pokoknya yang menggunakan jasa broker. Nah jadi sekian. Kalau untuk
pembahasan messaging. Jadi kalau saya teman teman ada kasus dimana integrasi servisnya butuh ngelakuin prosesnya asing kronos atau juga requestnya enggak perlu replay gitu ya enggak peduli ripe lainya atau respons seperti apa itu teman teman bisa implementasikan sie messaging. Nah, jangan lupa, teman teman enggak harus mengimplementasikan salah satu doang ya. Jadi
nggak harus implementasinya. Remote prosedur in, vocation atau messaging? Teman teman bisa kombinasikan. Contohnya. Kalau di kasus ini adalah kalau data email, sms, sodor

pakainya meses broker. Tapi kalau misalnya sore pakainya messaging. Tapi kalau misalnya nge get product itu langsung derek ke
Product servisnya menggunakan RESTful API Air misalnya. Jadi kan itu kombinasi antara remote, prosedur dan function dan juga messaging. Jadi teman teman bisa explore kemungkinan kemungkinan. Jadi ngga harus implementasi satu cara berkomunikasi. Teman teman juga bisa langsung informasikan dua cara komunikasi sekaligus dalam microservices.